

Easy computer-assisted modeling of dynamic systems

User's manual
for simulation system DYNAST

Herman Mann
Michal Sevcenko

Prague, Oct. 4, 2008

Contents

1	Examine DYNAST	1-1
1.1	What DYNAST can do for you	1-1
1.2	Try out DYNAST	1-2
1.2.1	Try DYNAST on the Internet	1-2
1.2.2	Try DYNAST on your computer	1-3
1.3	DYNAST availability	1-5
2	DYNAST software system	2-1
2.1	DYNAST Solver	2-1
2.2	DYNAST Shell	2-3
2.3	DYNAST distributed system	2-4
2.4	DYNAST files	2-5
2.4.1	File types	2-5
2.4.2	Folder association	2-6
2.4.3	Lists of problem and submodel files	2-7
2.4.4	DYNAST textual language	2-8
3	Function expressions	3-1
3.1	Function expressions	3-1
3.2	Variables and parameters	3-2
3.2.1	Numerical constants	3-2
3.2.2	Variable and parameter identifiers	3-3
3.3	Arithmetic and logical operators	3-3
3.3.1	Arithmetic operators	3-3
3.3.2	Logical operators	3-4
3.4	Standard and random functions	3-6
3.4.1	Standard functions	3-6
3.4.2	Random functions	3-6
4	User-defined functions	4-1
4.1	Inserting user-defined functions	4-1
4.2	Impulse functions	4-1
4.3	Tabular functions	4-2
4.3.1	Inserting tabular functions via dialog	4-2
4.3.2	Importing tabular functions	4-4
4.4	Polynomial functions	4-4
4.5	Text of user-defined functions	4-6

5	Altered functions and events	5-1
5.1	Insertion of altered functions	5-1
5.2	Transformed functions	5-2
5.3	Trimmed functions	5-3
5.4	Periodic functions	5-4
5.5	Text of altered functions	5-5
5.6	Events	5-5
6	Nonlinear equations	6-1
6.1	Equations and their variables	6-1
6.2	Submitting explicit equations	6-2
6.3	Submitting implicit equations	6-3
6.4	Text of nonlinear equations	6-6
7	Physical diagrams	7-1
7.1	Principles of multipole modeling	7-1
7.1.1	Approximating assumptions	7-1
7.1.2	Multipole models of dynamic systems	7-2
7.2	Variables of multipole models	7-3
7.2.1	Power and energy variables	7-3
7.2.2	Variable orientation	7-4
7.3	Automated formulation of equations	7-5
7.3.1	Postulate of continuity and compatibility	7-5
7.3.2	Constitutive relations of component models	7-6
7.3.3	Multipole modeling step-by-step	7-8
8	Physical elements	8-1
8.1	Variety of physical elements	8-1
8.1.1	Elements dissipating or accumulating energy	8-1
8.1.2	Sources of energy and related elements	8-3
8.1.3	Nonlinear, time-variable and controlled elements	8-4
8.1.4	Forbidden element configurations	8-6
8.2	Variables of physical elements	8-6
8.2.1	Power consumption of physical elements	8-6
8.2.2	Orientation of element variables	8-7
9	Block diagrams	9-1
9.1	Block diagram architecture	9-1
9.2	Basic blocks	9-2
9.3	Block diagrams and physical models	9-5
9.3.1	Block diagrams of physical models	9-5
9.3.2	Blocks in physical diagrams	9-6
10	Diagrams in graphical form	10-1
10.1	Creating diagrams	10-1
10.1.1	Environment for creating diagrams	10-1
10.1.2	Placing parts	10-2
10.1.3	Interconnecting parts	10-4

10.1.4	Diagram nodes	10-5
10.1.5	Using multilinks.	10-5
10.1.6	Inserting equations in diagrams	10-6
10.2	Editing, printing and exporting diagrams	10-6
10.2.1	Editing submodels from diagram window	10-7
10.2.2	Printing and exporting diagrams	10-7
11	Problems in textual form	11-1
11.1	Diagrams and their textual files	11-1
11.1.1	Synchronization of diagrams with problem text files	11-1
11.1.2	Submitting diagrams in textual form	11-2
11.2	Submitting texts of physical elements	11-4
11.2.1	Physical elements between nodes	11-4
11.2.2	Physical elements in series	11-5
11.2.3	Inductive coupling	11-5
11.3	Submitting texts of basic blocks	11-7
11.4	Submitting texts for submodel insertion	11-10
11.5	Catalogue of component parameters	11-12
12	Nonlinear analysis	12-1
12.1	Modes of nonlinear analysis	12-1
12.2	Submitting nonlinear analysis	12-2
12.2.1	Transient analysis	12-2
12.2.2	Desired variables of nonlinear analysis	12-3
12.2.3	Occurrence of desired-variable points	12-4
12.2.4	Static or steady-state analysis	12-4
12.2.5	Sweeping-parameter analysis	12-5
12.2.6	Initial conditions of transient analysis	12-6
12.2.7	Initial-solution estimate	12-7
12.2.8	Large-signal analysis	12-7
12.2.9	Changing or adding nonlinear analysis	12-8
12.2.10	Family of responses or characteristics	12-8
12.3	Fourier analysis	12-10
12.4	Text of nonlinear analysis	12-11
13	Numerical frequency analysis	13-1
13.1	Excitation for numerical frequency analysis	13-1
13.2	Submitting numerical frequency analysis	13-2
13.3	Text of numerical frequency analysis	13-4
14	Semisymbolic analysis	14-1
14.1	Options of semisymbolic analysis	14-1
14.1.1	Responses of block diagrams	14-1
14.1.2	Responses of physical diagrams	14-2
14.1.3	Responses in semisymbolic form	14-3
14.2	Submitting semisymbolic analysis	14-4
14.2.1	Submitting transforms of responses	14-4
14.2.2	Submitting time-domain responses	14-7

14.2.3	Submitting frequency characteristics	14-10
14.3	Text of semisymbolic analysis	14-12
14.3.1	Text of transform functions	14-12
15	Plotting results	15-1
15.1	Forms of plotting	15-1
15.1.1	Selection of variables	15-1
15.1.2	Plots in several windows	15-2
15.2	Scales of plots	15-2
15.2.1	Automatic setting of scales	15-2
15.2.2	Setting scales by the user	15-4
15.3	Arrangement of displayed plots	15-4
15.3.1	Graph grid and variable waveform marks	15-4
15.3.2	Notes in graphs	15-4
15.4	Reading waveform coordinates	15-5
15.5	Import, export and printing of graphs	15-6
15.5.1	Common plotting of different graphs	15-6
15.5.2	Plotting data from external sources	15-6
15.5.3	Printing and exporting plots	15-8
15.5.4	Saving plot layouts	15-8
16	Creating submodels	16-1
16.1	Submodel diagrams and equations	16-1
16.1.1	Submodel diagrams	16-1
16.1.2	Submodel equations	16-2
16.2	Submodel text files	16-3
16.2.1	Submodel text files from diagrams	16-3
16.2.2	Creating submodel dialogs	16-4
16.2.3	Creating submodel text from scratch	16-5
16.2.4	Submodel text from a problem text	16-5
16.3	Libraries of submodel symbols	16-7
16.3.1	Creating a new library	16-7
16.3.2	Basic symbol properties	16-7
16.3.3	Creating symbol patterns	16-8
16.3.4	Placing symbol pins	16-9
16.3.5	Editing symbol patterns	16-10
16.3.6	Editing symbol libraries	16-10
16.3.7	Exporting symbol libraries	16-11
16.4	Organization of submodel files	16-11
17	Modeling toolbox for MATLAB	17-1
17.1	DYNAST & MATLAB in control design	17-1
17.2	Exporting transfer functions to MATLAB	17-1
17.3	Controlling model in DYNAST by Simulink	17-2
17.3.1	Preparing the plant model in DYNAST	17-2
17.3.2	Required setup of MATLAB	17-3
17.3.3	Preparing the control diagram in Simulink	17-3

Chapter 1

Examine DYNAST

Chapter sections

1.1 What DYNAST can do for you	1-1
1.2 Try out DYNAST	1-2
1.3 DYNAST availability	1-5

Chapter overview. *Following this introductory chapter you can not only learn about DYNAST abilities, but also how you can examine these abilities right away on your computer, either online or offline. At the same time, you will find here DYNAST installation requirements and information about availability of its different versions.*

1.1 What DYNAST can do for you

DYNAST is able to simulate or analyze real dynamic systems the nonlinear dynamic models of which are represented by

- physical diagrams characterizing graphically the way in which the system is assembled from from real components without the need to formulate any equations (as DYNAST will formulate them automatically)
- sets of nonlinear algebro-differential equations in a textual form without constructing a block diagram
- block diagrams with very versatile blocks (including implicit ones) and any algebraic loops
- combinations of physical diagrams, blocks or equations

For system models submitted in such as way, DYNAST is able to compute

- transient responses to external excitations and initial conditions
- static or steady-state responses and their dependence on parameter variations
- responses to small or large signals in the vicinity of a quiescent operational point

- Fourier analysis of steady-state periodic responses
- linearized models in the vicinity of an operation point

For your linearized system models DYNAST can compute

- Laplace transforms of responses to external excitation and initial conditions in a semisymbolic form
- time-domain characteristics in a semisymbolic form
- coefficients or poles and zeros of transfer functions
- frequency characteristics of transfer functions with frequency-dependent coefficients

DYNAST is very useful also for control design as it allows for

- exporting transfer-function coefficients to the MATLAB environment
- co-simulation of digitally controlled systems with Simulink
- real-time control via Simulink


DYNAST exploits very efficient and robust algorithms in all these computation procedures.

1.2 Try out DYNAST

1.2.1 Try DYNAST on the Internet

If you will go to <http://virtual.cvut.cz/dyn/examples/>, you will find there

- numerous examples of re-solvable problems specified in terms of diagrams or equations
- submodels, i.e. models of various dynamic system components
- samples of interactive virtual experiments driven by DYNAST across the Internet

You can find examples or submodels of your interest by consecutive clicking the icon  in the left frame of the webpage.

Each example is presented on two webpages. The first one shows a problem assignment and the second one its solution. The problems can be re-solved by clicking the **Submit to DYNAST** button on the second page. The re-solving takes place on the Czech Technical University server in Prague. Before clicking the button, you may try to modify some of the problem parameters.

Some of the problem assignments are augmented with animated system models. The animations are driven by simulation results from the previous DYNAST execution. Note, that if you re-submit a modified problem you can observe either modified plots of systems responses, or a modified system animation.

At <http://virtual.cvut.cz/dyn/examples/> you can also try the DYNAST working environment DYNCAD in the form of an applet, that allows for schematic editing diagrams and plotting results without downloading any software.

1.2.2 Try DYNAST on your computer

Free download

of DYNAST is available at <http://virtual.cvut.cz/dyn/examples/>.


Installation requirements

include a PC computer with MS Windows 2000/NT/XP/Vista and about 50 Mbyte of free space on the hard disc.

Installation process

is very simple: execute the downloaded installation file and follow step-by-step the installation instructions.


Execute DYNAST Shell

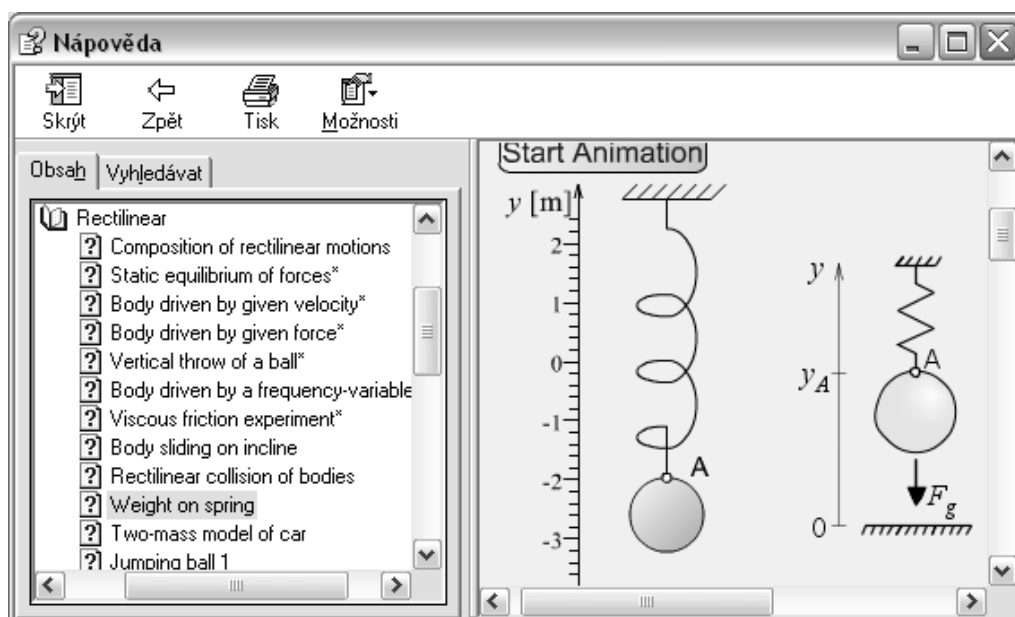
by clicking the icon 

Menus and the toolbar in the DYNAST Shell main window change according to the task you wish to accomplish and which working window you made active. You will thus not be unnecessarily distracted by tools which you do not need at the moment.

The easiest way of getting acquainted with the DYNAST Shell basic options is exploring solved examples stored in its Help system. To do this, choose Examples & Submodels in the Help menu.

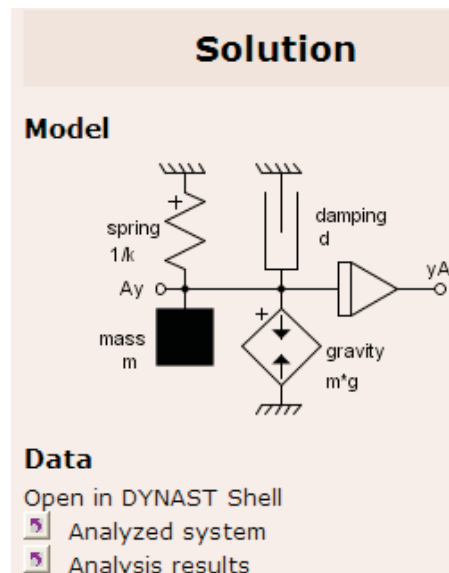
Choose an example

by consecutive clicking the icons  in the Contents tab. If you choose, for example, Mechanical and then the Rectilinear domain, you will open the Weight on spring example:



Browse through the solution section

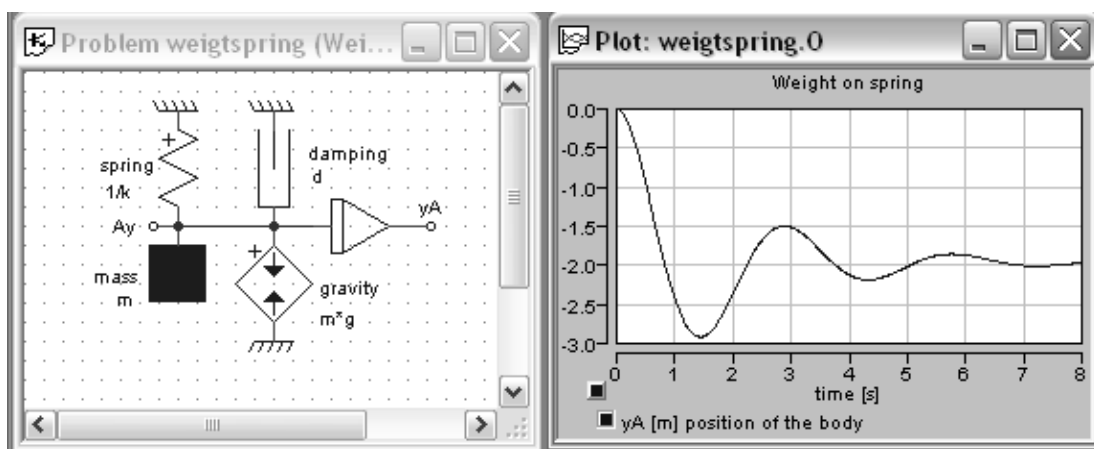
Let us assume that the point A in the above picture represents the y-motion of the weight. Then the point A projects onto the node A_y in the physical diagram below. The symbols linked to the node represent actions of the spring force, mass inertia and gravitational force on the point A of the real system. The fourth symbol in the diagram is a block integrating position y_A of the point A from the node velocity.



Click the button Analyzed system, to display the diagram in the window of the DYNAST Shell schematic editor.

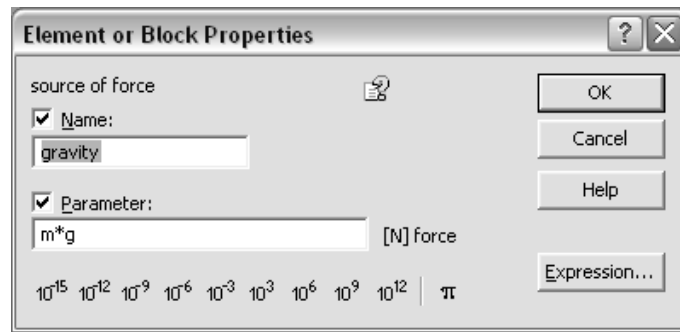
Execute analysis

If you click the Simulation Results button, DYNAST will start the analysis computation in addition. If your computer is connected to the Internet, still another window opens shortly after with a plot of the analyzed system response.

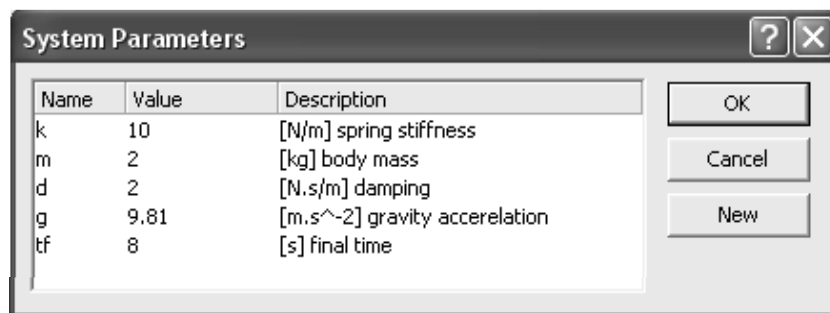



Modify the problem

Double click any of the symbols to open its dialog and change the value of its parameter. If you choose, for example, the force symbol, the following dialog will open:



If you would like to modify any parameter expressed symbolically like m , for example, choose Edit system parameters from the System menu to open the dialog:



Click now the  icon in the main-window toolbar to see the response for the modified parameter.

1.3 DYNAST availability

You can choose different versions of DYNAST Solver which provides number crunching for DYNAST Shell. The function of all the available solvers is the same, they differ only in the maximum number of equations they can solve and in some solving procedures.

To select a suitable solver, choose Options in the Preferences menu in DYNAST Shell, and then choose the solver in the Solver tab.

- Solver online allows solving problems across the Internet on our server in Prague without any limitation of problem size.
- Firewall-immune Solver online is similar to the previous solver, but it was designed for users with a limiting access to the Internet due to a fire wall.
- Lite Solver offline can be used without an Internet access, but for problems of a rather limited size only.
- Student Solver offline allows solving larger problems than Lite Solver, but requires a soft key obtainable after user's registration at www.virtual.cvut.cz/cgi-bin/Register/
- Professional Solver offline allows solving problems offline without any limitations to users who asked for a hardkey at dyn@virtual.cvut.cz



Solver
selection.

Users with a large number of clients like schools, for example, can install DYNAST Solver on their own server with MS Windows. DYNAST will be then accessible to their clients via the local network.

Chapter 2

DYNAST software system

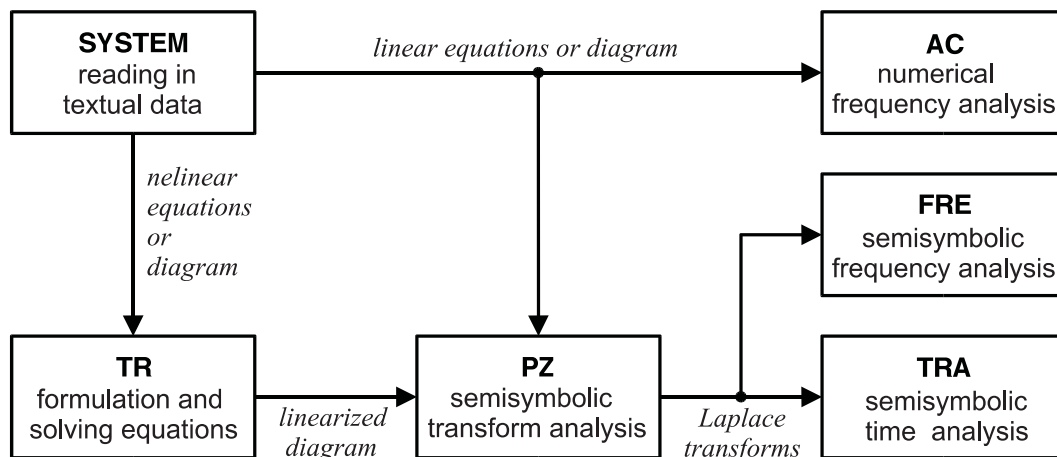
Chapter sections

2.1	DYNAST Solver	2-1
2.2	DYNAST Shell	2-3
2.3	DYNAST distributed system	2-4
2.4	DYNAST files	2-5

Chapter overview. *This is a description of the main parts of the DYNAST software system and their mutual communication. It gives a survey of DYNAST major functions and of the file types necessary for implementing these functions.*

2.1 DYNAST Solver

DYNAST Solver makes the core of the DYNAST software system. It does all the necessary equation formulation and solution required by the submitted problems and it performs their simulation or analysis. DYNAST users, however, do not communicate with DYNAST Solver directly. For submitting their problems to DYNAST and for interpretation of results they use DYNAST Shell, the software working environment.



The blocks in the above figure represent the fundamental sections of the DYNAST Solver, and the arrows between the blocks indicate the data flow through DYNAST Solver. Table 2.1 gives a survey of functions of the individual sections.

Table 2.1: Overview of DYNAST sections.

SECTION	PURPOSE
SYSTEM	Reading in models in the form of <ul style="list-style-type: none"> • systems of algebro-differential equations • physical diagrams • block diagrams • combinations of these
TR	Transient analysis and its special cases like <ul style="list-style-type: none"> • static analysis • steady-state analysis • parameter swept analysis • detection of event occurrence • determination of quiescent operation point • linearization of analyzed systems • Fourier analysis of periodic steady state
PZ	Semisymbolic analysis of Laplace transforms <ul style="list-style-type: none"> • of transfer functions • of responses to initial conditions
TRA	Semisymbolic and numerical analysis of time-domain <ul style="list-style-type: none"> • transfer-functions characteristics • system responses to initial conditions
FRE	Frequency characteristics of semisymbolic transfer functions
AC	Numerical frequency analysis of frequency-dependent diagrams

The problems submitted by users either in a graphical or textual form are converted in DYNAST Shell to textual data. These data are read in by the SYSTEM section of DYNAST Solver and used for preprocessing equations underlying the submitted problems.

The linear problems can be directly analyzed numerically in the frequency domain in the AC section, or characterized by Laplace transforms in the PZ section. The former analysis requires numerical solution of systems of complex-coefficient equations. The latter operation exploits the double eigenvalue problem solution.

The resulting rational functions of the Laplace-transform variable s can be inverted in the TRA section to obtain semisymbolic or numerical form time-domain responses. The transform functions can be also converted into their frequency characteristics in the FRE section.

Systems of nonlinear equations submitted by the user, or derived automatically by DYNAST for a submitted diagrams, are solved simultaneously in the implicit form in the TR section. They are solved by the same algorithm regardless whether they are differential, algebraic, or an algebro-differential mixture. There is only one numerical equation-solving procedure available in DYNAST, but it is very efficient and computationally robust. This relieves DYNAST users from the ambiguous decision which procedure to choose.

Both the step size and order of the numerical procedure are continuously optimized during the solution process to minimize the computation time while respecting the permitted accuracy tolerance. To increase the computation efficiency further, symbolic differentiation to evaluate equation jacobians and sparse-matrix techniques are utilized. Even if the user-specified initial conditions of the solution are inconsistent, DYNAST is able to find their nearest consistent values.

Fast Fourier transform is used to compute frequency spectra of steady-state responses.

2.2 DYNAST Shell

DYNAST Shell provides a user-friendly access to DYNAST Solver. It has been designed to suit well both to newcomers and advanced users. All operations are intuitive and supported by a **context-sensitive help** system. Simultaneous pressing of the Shift and F1 keys opens a help message related to the tool you are currently using. A built-in **syntax analyzer** is continuously checking the submitted data and gives an **error message** in case of its occurrence.

The system of menus placed at the top of the main window as well as the variety of icons at the toolbar below it do not remain unchanged during the DYNAST Shell exploitation. The **menus and toolbar change** depending on which of the working windows is currently active. This simplifies considerably users' orientation in the tools and options available for different operations.

The graphical user interface of DYNAST Shell formed by dialog windows (wizards) allows for submitting problems and their solution without learning any special language. The input data is directly interpreted without any compilation delay.

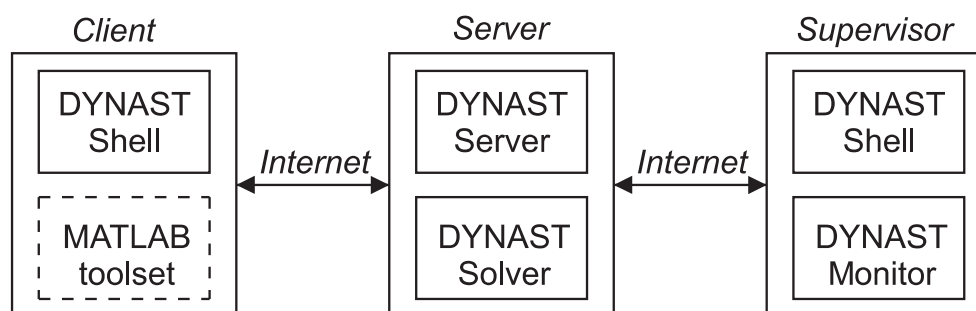
The preferred way of submitting problems to DYNAST Shell is the graphical one. Users set up their physical or block diagrams from symbols representing physical phenomena, real components or mathematical relations. The symbols are stored in symbol libraries which are open in the sense that users may freely create and add submodels and symbols of their own. Built in the program, there is both a schematic editor for diagrams and a graphical editor for creating symbols. A special dialog box for each new submodel symbol is formed in DYNAST Shell automatically.

Results of problems solved in DYNAST Solver are returned back to DYNAST Shell in a textual form. Various responses and relations can be plotted with automatic scaling. DYNAST Shell includes also a software tool for animation of analyzed systems that change their shape or color. The software allows currently for display of 2D animations, 3D animations should be displayed by a third-part browser.

Both the problem specifications and their results can be documented using a built-in documentation system using standard templates. The systems extracts the relevant parts of the input data and captures diagrams as well as the resulting plots and includes them into the documents. These can be converted to PDF or HTML formats.

2.3 DYNAST distributed system

DYNAST Shell can communicate with DYNAST Solver not only off line, but also online across the Internet or a local network. This option is illustrated by the following figure.



Such a **distributed system** can be composed of many client computers the users of which have their tasks processed by a single DYNAST Solver implemented together with DYNAST Server on a common server computer.

In addition, activities between clients and the server can be supervised from another remote computer. The software tool called **DYNAST Monitor** allows for supervising the data submitted to the server both in the textual and graphical forms. The supervisor can not only monitor clients' activities, but also communicate with them by e-mails to assist them in solving their problems and correct their errors if necessary.

The clients are supported by a large collection of re-solvable examples and component sub-models as well as by a course on physical modeling. The **DYNAST Manager** allows for automated administration of such an amount of files, for their conversion to the HTML and PDF formats, and for uploading them to the server.

Clients with the **MATLAB toolset** installed can import to it semisymbolic-form transfer functions from DYNAST exploiting thus the advantage of its higher modeling efficiency over MATLAB. On the other hand, parameters of the DYNAST models can be optimized by the MATLAB **optimization** toolbox. Digital control of models in DYNAST can be provided by **co-simulation** with Simulink implementing the digital-control configuration. Also **real-time control** can be executed via Simulink. In all these operations, the MATLAB toolset can communicate with DYNAST solver either offline or online.

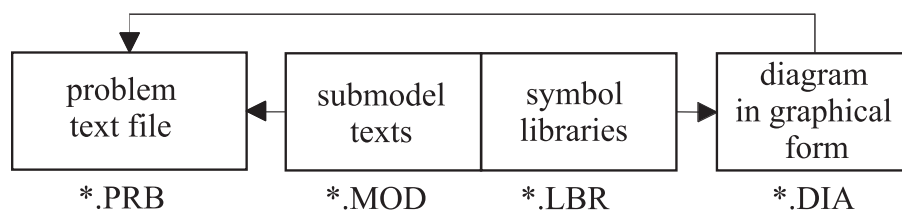
2.4 DYNAST files

2.4.1 File types

Table 2.2: DYNAST file type extensions

EXTENSION	FILE	DESCRIPTION
<i>problem specification</i>		
DIA	system diagram	diagram in graphical form
PRB	problem text	text of system model and its analysis
O	result text	text of analysis results
INIT	initial conditions	initial conditions of nonlinear analysis
FTN	tabulated function	text with tabulated function values
LAY	window layout	layout of windows in DYNAST Shell
<i>submodel specification</i>		
LBR	symbol library	library of submodel graphical symbols
DIA	submodel diagram	diagram of submodel in graphical form
MOD	submodel text	text of submodel specification
CAT	parameter catalogue	catalogue of submodel parameters

Table 2.2 gives types and extensions of data files processed by DYNAST Shell. The following figure indicates the relationships among the four basic file types used by DYNAST to solve problems.



The PRB files are indispensable for solving any problems by DYNAST even though users need not see them in most cases. For problems submitted in the graphical form of diagrams the related PRB files are generated automatically by DYNAST. Each PRB file consists of two parts: textual description of the submitted diagram (netlist) or equations, and specification of the required analyses. Users usually use the textual-form PRB files only for submitting their problems specified by equations.

Diagram schematics are stored in DIA files. They are set up from graphical symbols organized in symbol libraries which are stored in LBR files. Symbols of submodels, i.e. models of dynamic system components, are accompanied by independent textual MOD files specifying the submodel dynamic behavior. When the problem specified by a diagram is submitted for

analysis, DYNAST first converts the diagram into a textual FRB file. Then, for each submodel symbol found in the diagram, DYNAST includes the related MOD file into the PRB file.

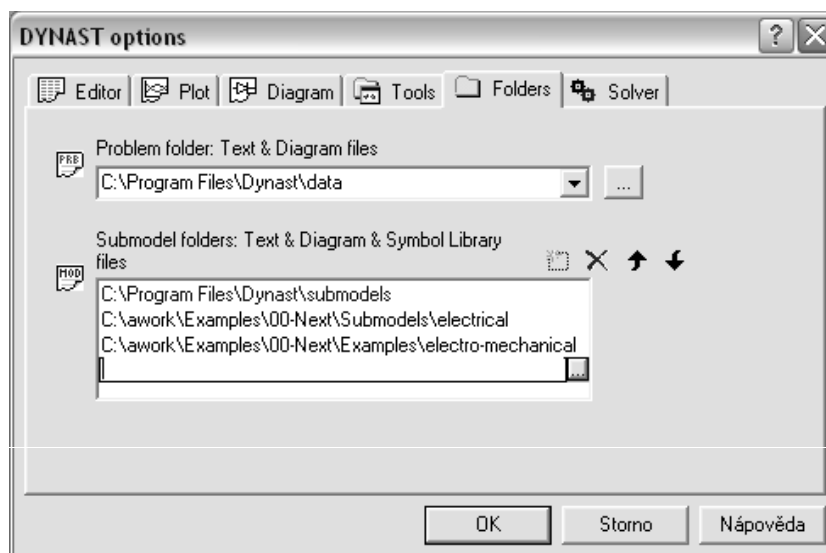
Analysis results are stored in tabular form in O files. The graphs stored in tables can be plotted by DYNAST Shell in different forms. Though the plots cannot be stored individually, they can be stored as parts of the complete DYNAST Shell window layout in LAY files. The individual plots can be converted to different formats and printed or exported.

As shown in Table 2.3, to open some of the mentioned files you can choose commands available in the **View** menu or you can click the corresponding icons in the toolbar. The other menu commands and icons available in DYNAST Shell are commented in the next chapters.

Table 2.3: Tools for viewing files.

ICON	COMMAND IN MENU <u>V</u> IEW	OPENS
	List of Problems	list of PRB and DIA files in associated folder
	List of Submodels	list of submodel MOD files in associated folders
	Problem text	problem text PRB file for active diagram
	Diagram	diagram for active problem text PRB file
	Result text	result O file for active problem text or diagram
	Result plot	result plots for active problem text or diagram

2.4.2 Folder association



Association
of working
folders.

Files with problem specification can be created in any folder of your computer. However, to exploit all the user comfort DYNAST Shell offers, you should properly associate folders you use for your work with the program.

The dialog for **folder association** opens, if you choose **Options** from the **Preferences** menu. Open then the **Folders** tab and set your working folder in the **Problem folder** box. DYNAST Shell will store in this folder all DIA files with diagram schematics, all problem text PRB files, all the related result O files, and also all the layout LAY files.

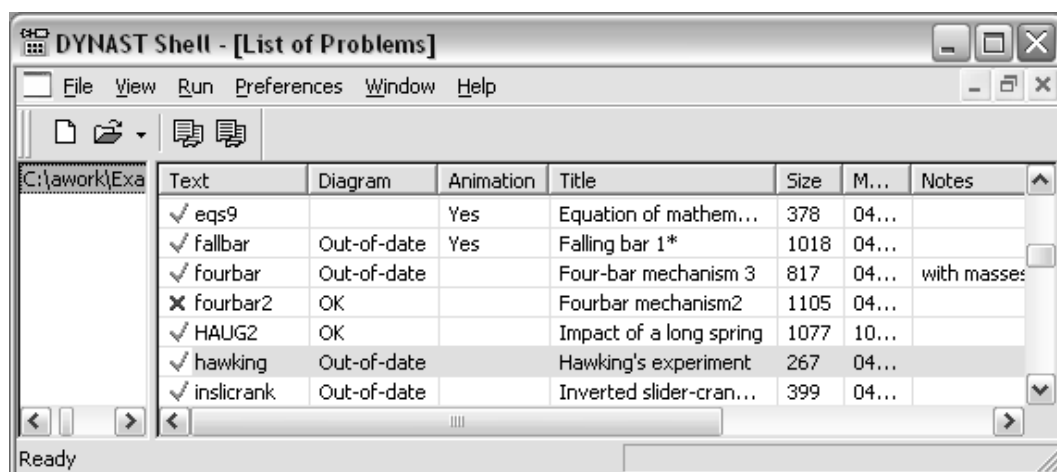
At the same time, you can set in the Submodel folders box several folders for storing submodel MOD text files. The first of the folders is fixed, however, as it stores the standard submodels coming from the DYNAST installation. For safety reasons, you should not store your own submodels in this folder. Avoid namely any modifications of the standard submodels as your modifications might be lost after upgrading your DYNAST installation.

The order of the associated folders determines the way in which the submodel files are searched for (this is important in case of different files with the same name). DYNAST searches through the associated folders in the following order:

1. Folder with the PRB file calling the submodel and its subfolders.
2. Folder with the standard submodels and its subfolders.
3. The other folders and their subfolders.

2.4.3 Lists of problem and submodel files

If you want to see which problems are stored in the associated working folder and what is their status, choose Problem List from the View menu.



The green OK characters in the first column inform you that there are no syntactic errors in the related problem files. The red cross indicates syntactic errors either in the PRB file, or in some of the associated MOD files. The PRB file opens in DYNAST Shell if you click it in the list and the error statement will be underlined in red color there. When you place the mouse cursor over the underlining a clarifying **error message** will appear at the state row of the DYNAST Shell window.

OK message in the Diagram column indicates that there is a DIA file in the working folder corresponding exactly to the related PRB file. No message means that there is no related DIA file in the folder. The out-of-date message indicates that the DIA file is present, but it is not fully compatible with the PRB file. If you click the out-of-date message, the diagram from the DIA file will open in DYNAST Shell. After correcting it select its row in the List of Problems and press the F5 key to initiate the DIA **file verification**. If the result is positive, the OK message appears.

If you select a row in List of Problems and press the Del key, the PRB file and all files associated with it will be deleted from the working folder.

You can open a similar list of all submodel MOD files stored in the associated folders and their subfolders. Just choose List of Submodels from the View menu.

2.4.4 DYNAST textual language

DYNAST was designed primarily to provide graphical communication with its users with a textual language underlining only the graphical communication. Yet, in some cases, using directly the DYNAST textual language may be helpful. In any case, this language is strongly problem oriented, very versatile and simple to use.

DYNAST textual language is composed of statements coded in ASCII characters. Both upper and lower case letters may be used, but DYNAST is **case insensitive**. Each statement is terminated by the semicolon character ;. A statement may continue on several lines, and there may be several statements in one line. The **statements** consist of keywords, identifiers, and user-defined identifiers. These items are separated by delimiters made by characters which are neither letters nor numbers, i.e., by characters like /, -, =, , and ., or by spaces. The underscore character _, however, is treated as a letter. It is fully insignificant if only one or more subsequent spaces are used as a delimiter. The **user-defined identifiers** may not contain any non-alphanumeric characters or spaces.

If there is a colon : in a line, the part of the line to the right of this character is considered as a **comment** and ignored as such by the program. (This feature may be used for deactivating some statements during the input data debugging.) When the colon : is preceded by the asterisk *, the comment will be used by DYNAST as a **problem title** in all related files. The notation used in statement definitions in some of the following chapters is given in Table 2.4.

Table 2.4: Notation used in statement definitions.

NOTATION	MEANING
<i>slanted style</i>	keywords or reserved identifiers
typewriter style	examples of input data
<i>italic style</i>	user-specified identifiers
statements between []	optional statements that can be omitted

The complete configuration of the PRB problem text files:

```
*SYSTEM;
data pro section SYSTEM
*section;
data for next section
*section;
...
*END;
```

Each section of the PRB file begins with the statement **section*; where *section* is the identifier of the section. The list of section identifiers is given in Table 2.1. The statement **END*; encloses the PRB file.

Chapter 3

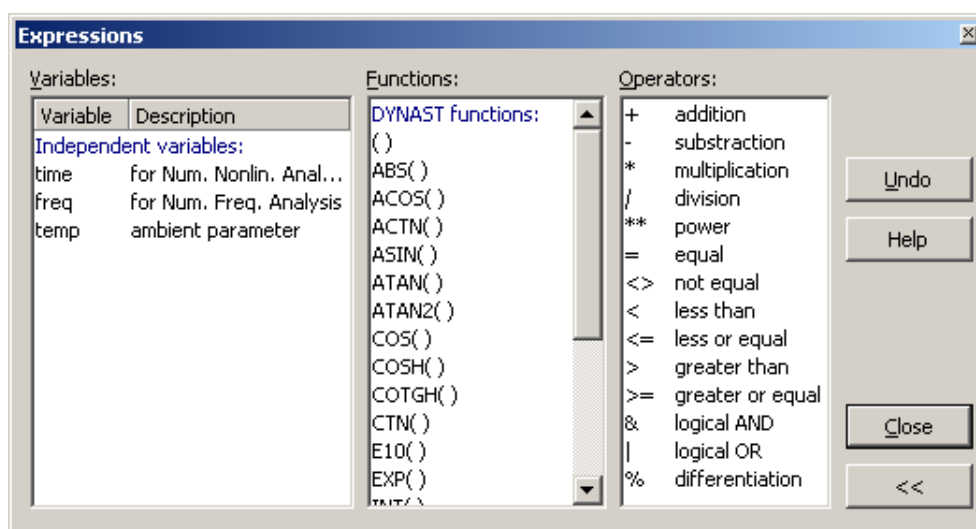
Function expressions

Chapter sections

3.1	Function expressions	3-1
3.2	Variables and parameters	3-2
3.3	Arithmetic and logical operators	3-3
3.4	Standard and random functions	3-6

Chapter overview. You will learn here in which formats to enter various variables or parameters and arithmetic or logic expressions in a symbolic form into submitted equations, physical elements or blocks. For inserting more complex functions or events see Chapters 4 and 5.

3.1 Function expressions



Inserting variables, operators and functions.

Function expressions can be easily entered using the Expressions dialog. It can be opened either from the menu System by selecting Insert Expression, or from dialogs for submitting equations, blocks, or physical elements. The Expressions dialog lists operators and functions that you have at your disposal. It also shows the variable and parameter identifiers which you

have used already in your current problem specification. This list is automatically updated whenever you submit a new statement.

3.2 Variables and parameters

3.2.1 Numerical constants

Constant numerical values of variables or parameters can be specified either as

$$\textit{fraction} [\textit{E exponent}] \quad \text{or as} \quad \textit{fraction} [\textit{suffix_unit}]$$

fraction is a fractional portion of the numerical constant. The decimal point can be placed in any position in it. If the value of the fractional portion is integer, the decimal point may be omitted.

exponent specifies an integer number n such that 10^n is within the range of the computer arithmetics.

suffix can be a **scale suffix** the variety of which is given in Table 3.1. There must not be any space between the number fractional portion and its scale suffix.

unit can be a **unit suffix**, i.e., an arbitrary string of alphanumerical characters which is separated from the preceding scale suffix or fractional portion by the underscore character `_` without any space. The unit suffix is ignored by DYNAST.

Table 3.1: Scale suffixes.

SUFFIX	SCALE FACTOR	VALUE
T	tera	10^{12}
G	giga	10^9
ME	mega	10^6
K	kilo	10^3
M	mili	10^{-3}
U	micro	10^{-6}
N	nano	10^{-9}
P	pico	10^{-12}
F	femto	10^{-15}
PI	Ludolphian number	π

Examples. The following numerical constants are all legal numbers:

```
-3.4 .3 67.08E-10 6.3K 5N 1K_VOLT 100_OHM -1PI
```

Pay attention to the fact that `PI` is a scaling factor. This means that it must be always preceded by a number without any space. Note also, that the string `1_FARAD` is interpreted as the numerical constant 1.0, while the string `1FARAD` is understood by DYNAST as 10^{-15} , i.e., as 1 femto.

3.2.2 Variable and parameter identifiers

Table 3.2 lists the types of variables and parameters which can be used in DYNAST expressions. Values of variables *TIME* and *FREQ* are controlled by DYNAST automatically within the user-specified range. The character string *V.* preceding a node identifier can be omitted if the first character in the node identifier is a letter, not a number.

Further details about the variables and parameters are given in chapters on submitting equations (Chapter 6), blocks (Chapter 9), physical elements (Chapter 8) and submodels (Chapter 16).

Table 3.2: Variables and parameters.

FORMAT	VARIABLE OR PARAMETER	BY DEFAULT
INDEPENDENT		
<i>TIME</i>	global variable of transient analysis	0
<i>FREQ</i>	global variable of frequency analysis	0
<i>TEMP</i>	global parameter	300
SOLVED		
<i>variable</i>	variable of an implicit equation	
<i>VD. variable</i>	numerical derivative of an implicit-equation variable	
<i>V. node</i>	node (across) variable	
<i>VD. node</i>	numerical derivative of node (across) variable	
<i>I. element</i>	through variable of elements R, L, E, S, O	
<i>ID. element</i>	numerical derivative of element R, L, E, S, O through variable	
EVALUATED		
<i>variable</i>	variable or parameter defined by explicit equation	1 10 ³⁰⁰
<i>V. element</i>	across variable of elements R, L, E, S, O	
<i>I. element</i>	through variable of elements G, C, J	
<i>parameter</i>	element parameter	
<i>event</i>	event variable	

3.3 Arithmetic and logical operators

3.3.1 Arithmetic operators

The DYNAST variety of arithmetic operators is shown in Table 3.3. The operator % for symbolic differentiation is of the lowest priority. It must be placed as the last operator in an expression. If only a part of an expression should be differentiated, the part must be placed between parentheses ().

Note the following differences between symbolic and numerical differentiation:

- statement *%TIME* denotes symbolic differentiation with respect to the independent variable *t* in the symbolic expression preceding this statement
- the statement *VD. variable* stands for the *variable* differentiated with respect to *t* numerically by DYNAST

Table 3.3: Arithmetic operators.

OPERATOR	PRIORITY	OPERATION
**	5	power
*	4	multiplication
/	4	division
+	3	addition
-	3	subtraction
%	0	symbolic differentiation

Examples. The following examples show arithmetic relations and their expressions in the DYNAST language.

$$x = k \frac{\sqrt{a^2 + b^2}}{2} \quad x = k*(a**2 + b**2)**(1/2)/2;$$

$$y = \frac{d}{dt} \left(\frac{t}{1-t^3} \right) \quad y = \text{time}/(1 - \text{time**3})\% \text{time};$$

$$p = \frac{d}{dx} (x^3 - 2x) \quad p = x**3 - 2*x \%x;$$

$$s = x \frac{d}{dx} \left(\frac{x^2}{x+1} \right) + x \quad s = x*(x**2/(x+1)\%x) + x;$$

$$z = \frac{\partial^2}{\partial x \partial y} \left(\frac{x^2 + y^2}{x} \right) \quad z = (x**2 + y**2)/x\%x\%y;$$

3.3.2 Logical operators

The variety of **logical operators** available in DYNAST is given in Table 3.4. If a logical expression is true, its numerical value is 1, if it is false, its value is 0. If an arithmetic expression of a numerical value $n > 0.5$ is inserted into a logical expression it is considered as true, otherwise it is taken as false.

Table 3.4: Logical operators.

OPERATOR	PRIORITY	OPERATION
~ or '	6	logical NOT
<	2	less than
>	2	greater than
<=	2	less or equal
>=	2	greater or equal
<>	2	not equal
=	2	equality
&	1	logical AND
	1	logical OR

Examples. You can submit the logical expression $X = (A \text{ OR } B) \text{ AND } (C \text{ OR } D)$ as

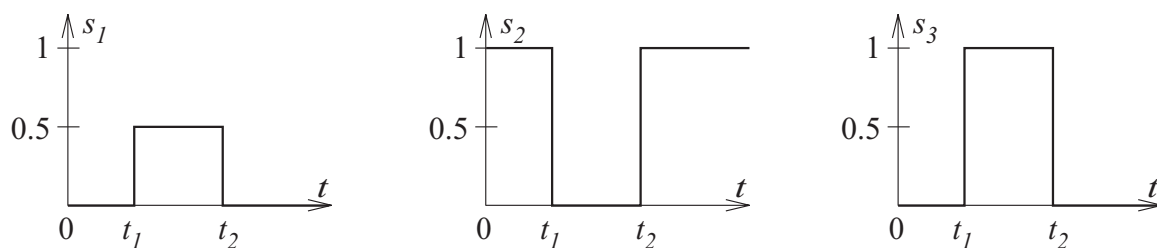
$$X = (A \mid B) \& (C \mid D);$$

Using logical expressions you can also submit **branched expressions**. For instance, the expression

$$y = \begin{cases} 10 & \text{for } z \leq 2 \\ 3z + 4 & z > 2 \end{cases}$$

can be submitted as

$$Y = 10 * (Z \leq 2) + (3 * Z + 4) * (Z > 2);$$



The figure illustrates DYNAST interpretation of the following expressions:

$$s1 = 0.51 * (\text{time} > t1) \& (\text{time} < t2); \quad s2 = \sim s1; \quad s3 = s1 \<> 0;$$

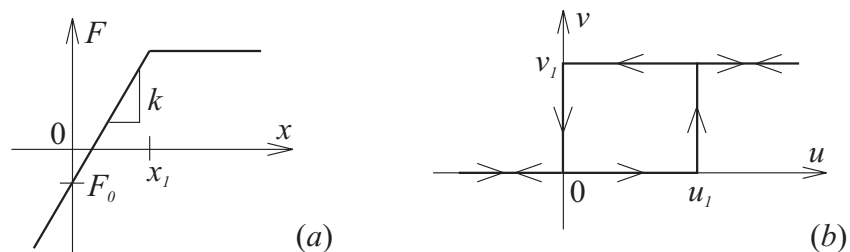


Fig. *a* gives an example of a piece-wise nonlinear characteristic submitted as

$$F = k * x * (x < x1) + k * x1 * (x \geq x1) + F0;$$

The characteristic shown in Fig. *b* exhibits hysteresis so that the values of v are controlled not only by values of u , but also by the sign of their changes. Using logical operators, v can be expressed by as

$$v = v1 * ((u \geq u1) \mid ((u > 0) \& (\forall D.u < 0))) ;$$

3.4 Standard and random functions

3.4.1 Standard functions

The complete list of DYNAST standard functions allowed in the functional expressions is given in Table 3.5.

Table 3.5: Standard functions.

TYPE	FUNCTION	TYPE	FUNCTION
ABS	absolute value	LOG	natural logarithm
EXP	exponential	LOG10	decadic logarithm
SIN	sine	SQRT	square root
COS	cosine	INT	integer part
TAN	tangents	SGN	signum
ATAN	arcus tangents	CTN	cotangens
SINH	hyperbolic sine	ASIN	arcussinus
COSH	hyperbolic cosine	ACOS	arcuscosinus
TANH	hyperbolic tangents	ACTN	arcuscotangents
E10	decadic exponential	COTGH	hyperbolic cotangents

Examples. Mathematical relations and the corresponding expressions in the DYNAST language:

$$x = a \cdot \sin(b^2 + 3)$$

$$X = A * \sin(B^{**2} + 3);$$

$$y = \frac{3}{\sqrt{a^2 + 10\cos 5(a+b)^2}}$$

$$Y = 3/\text{SQRT}(A^{**2} + 10*\text{COS}(5*(A + B)^{**2})) \quad ;$$

$$z = \frac{\partial^2}{\partial x \partial y} x^3 e^{\tan y}$$

$$Z = X^{**3} * \text{EXP}(\text{TAN}(Y)) \% X \% Y;$$

3.4.2 Random functions

The time-dependent random function $r(t)$ generated by DYNAST is of the normal or uniform distribution. The random-function specification is either

$$\text{normal}(\text{mean value}, \text{variation}, \text{delta}) \quad \text{or} \quad \text{uniform}(\text{lower}, \text{upper}, \text{delta})$$

The random function is based on random numbers generated at equidistant points of the variable `TIME` separated from each other by the interval *delta*. The random function values between these points are given by linear interpolation similarly as in the tabular function. All parameters – *mean value*, *variation*, *lower*, *upper*, *delta* – should be numerical constants or constant expressions.

Example. Random numbers of normal distribution with mean value equal to 15 and with variation equal to 0.8 generated at time points displaced by 10 ms can be submitted as

$$A = \text{NORMAL}(15, 0.8, 10\text{m});$$

Chapter 4

User-defined functions

Chapter sections

4.1	Inserting user-defined functions	4-1
4.2	Impulse functions	4-1
4.3	Tabular functions	4-2
4.4	Polynomial functions	4-4
4.5	Text of user-defined functions	4-6

Chapter overview. *DYNAST allows you to define some useful non-standard functions, which can be used with arbitrary arguments. Each function definition must precede the first expression in which it has been used. Tabular functions can be user-defined or imported from external files (generated by measuring instruments, for example).*

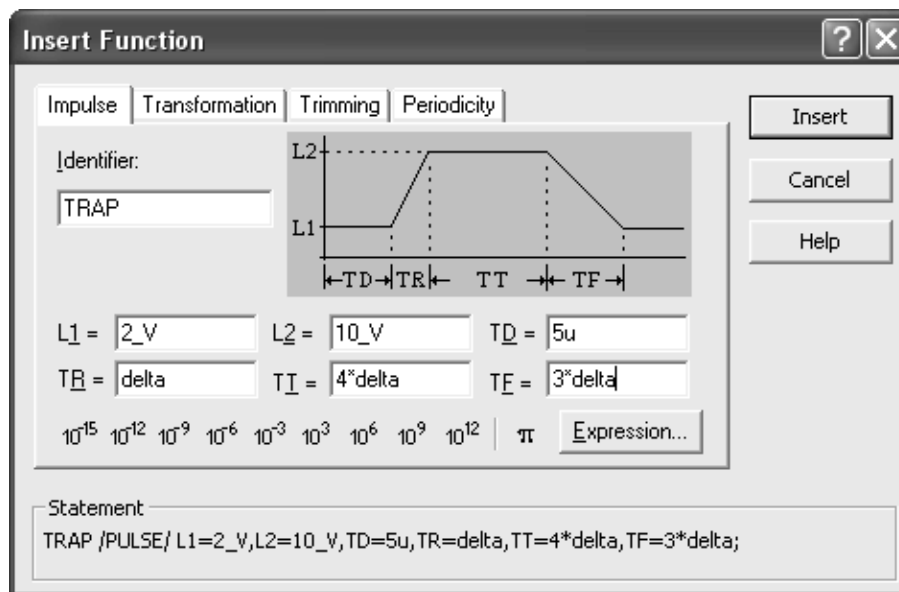
4.1 Inserting user-defined functions

Dialogs for inserting user-defined functions can be opened from the System menu. The function can be inserted in these steps:

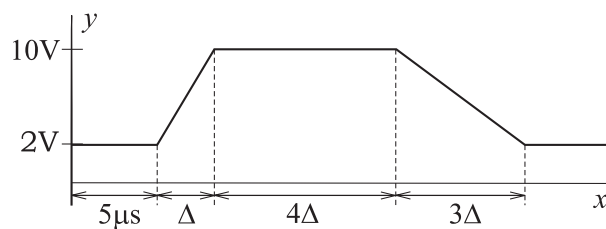
1. Choose an identifier for the submitted user-defined function $f(\bullet)$. The symbol \bullet represents an arbitrary argument.
2. Specify values for the function parameters according to instructions given in the following paragraphs.
3. The submitted function can be then transformed, trimmed or made periodic using the dialog tags (Chapter 5).

4.2 Impulse functions

To open the dialog for inserting an impulse functions choose Insert Impulse Function in the System menu.



Inserting
impulse
functions.



Example. The impulse function plotted above, denoted by the identifier `TRAP`, can be entered as shown in the dialog screenshot. This results in the automatically specified statement

```
TRAP /PULSE/ L1 = 2_V, L2 = 10_V, TD = 5u, TR = delta,
TT = 4*delta, TF = 3*delta;
```

Note, that the form of some parameter values is symbolic as they are specified as multiples of the variable `delta`.

Such function can be then used with different arguments as shown in the following equations

```
v1 = TRAP(time - t0); y2 = TRAP(x1**2);
```

4.3 Tabular functions

4.3.1 Inserting tabular functions via dialog

DYNAST allows you also submitting a function $y = f(x)$ in the form of a sequence of points specified by couples of their coordinates (x_i, y_i) . For this purpose, you can use the dialog opened by selecting Insert Tabular Function from the System menu.

Insert Function

Tabular Transformation Trimming Periodicity

Identifier: DIO1

Points:

Argument	Function
-1k	-10
0	0
30	1k

Insert Delete

10⁻¹⁵ 10⁻¹² 10⁻⁹ 10⁻⁶ 10⁻³ 10³ 10⁶ 10⁹ 10¹² π Expression...

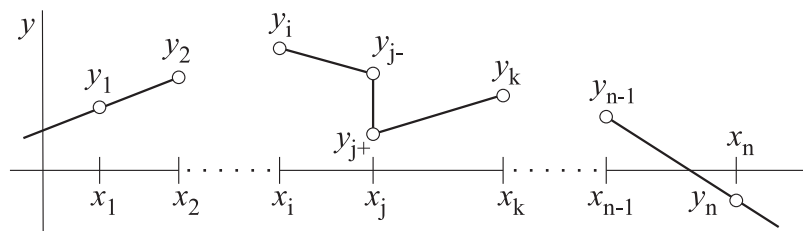
Statement

DIO1 /TAB/ -1k,-10, 0,0, 30,1k;

Insert Cancel Help

Inserting
tabular
functions.

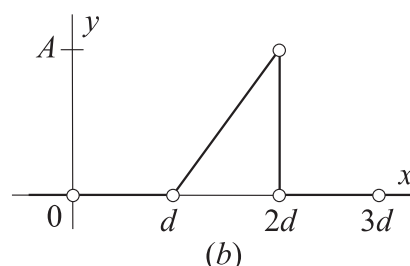
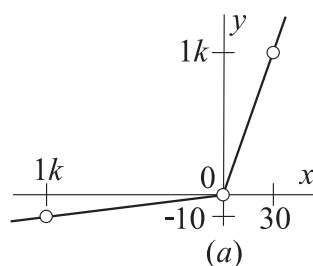
The values of arguments x_i and the corresponding functions y_i , $i = 1, 2, \dots$, can be numeric constants or symbolic expressions. The argument values should be submitted in the ascending order, so that $x_i \leq x_{i+1}$ for each i . A function discontinuity $y_{i-} \neq y_{i+}$ at x_i , should be specified by two value couples: x_i, y_{i-} and x_i, y_{i+} .



This figure illustrates the way in which DYNAST evaluates tabular functions between their submitted discrete points. DYNAST interpolates the function $y = f(x)$ at any point x between neighboring arguments x_i and x_{i+1} (where $x_i < x < x_{i+1}$) by a line segment such that

$$y = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} (x - x_i) + y_i$$

Outside the interval $\langle x_1, x_n \rangle$ defined by the table, the function y is extrapolated linearly. For any $x < x_1$, the function y is extrapolated by a line passing through the first two table points at x_1 and x_2 . Similarly, for any $x > x_n$, the function y extrapolates a line passing through the last two points at x_{n-1} and x_n .



Examples. Parameters of the function given in Fig. *a* are shown specified in the dialog screenshot. Their submitting will result in the statement

```
DIO1 /TAB/ -1k,-10, 0,0, 30,1k;
```

The saw-tooth impulse function shown in Fig. *b* can be specified by as

```
GAP/TAB/ 0,0, d,0, 2*d,A, 2*d,0, 3*d, 0;
```

These functions can be utilized, for example, in equations

```
i = DIO1(v);    Z = GAP(time);
```

4.3.2 Importing tabular functions

You may also import tabular user-defined functions via ASCII files. This option enables reading in function relationships from digital measuring instruments, for example.

The tabular function should be stored in the file

file.FTN

where *file* stands for the file name. The file should contain couples of point coordinates in the form of numeric constants separated by commas or spaces.

The statement for importing the point coordinates from the file should be in the form

FILE = *file*;

Example. The tabular user-defined function DIO1 stored in the file FD.FTN in the form of data

```
-1k -10    0 0    30 1k
```

could be later imported using the statement

```
DIO1 /TAB/ FILE=FD;
```

4.4 Polynomial functions

A real polynomial function

$$f(x) = a_0 + a_1x + a_2x^2 + \dots = k(x - x_1)(x - x_2) \dots$$

can be submitted either in terms of its coefficients or roots. You can open the dialogs for either of these options by selecting Insert Polynomial from the System menu.

Insert Function

Coefficients | Composed | Trimmed | Periodic

Identifier: $f(x) = a_0 + a_1x + \dots + a_nx^n$

Coefficient	Value
a0	0
a1	4
a2	0
a3	1

10⁻¹⁵ 10⁻¹² 10⁻⁹ 10⁻⁶ 10⁻³ 10³ 10⁶ 10⁹ 10¹² | π Expression...

Statement
F /POLY/ 0,4,0,1;

Buttons: Insert, Cancel, Help, Insert, Delete

Inserting
polynomial
functions
in terms of
coefficients.

Both coefficients and roots including the factor k can be specified in terms of numeric constants or symbolic expressions. The polynomial order is determined automatically by DYNAST based on the number of submitted coefficients or roots.

If the submitted real polynomial has a pair of complex conjugate roots

$$x_k = \text{Re } x_k + j \text{Im } x_k, \quad \bar{x}_k = \text{Re } x_k - j \text{Im } x_k$$

it is sufficient if you submit only one of them.

Insert Function

Roots | Composed | Trimmed | Periodic

Identifier: $f(x) = k(r_1 + js_1)(r_2 + js_2) \dots (r_n + js_n)$

Multiplicative constant:

Root real part	Root imaginary part
0	0
0	2
0	-2

10⁻¹⁵ 10⁻¹² 10⁻⁹ 10⁻⁶ 10⁻³ 10³ 10⁶ 10⁹ 10¹² | π Expression...

Statement
F /ROOT/ 1,[0,0],[0,2],[0,-2];

Buttons: Insert, Cancel, Help, Insert, Delete

Inserting
polynomial
functions
in terms of
roots.

Examples. The dialog screenshots show inserting the polynomial

$$y = 5x^3 + 20x = 5x(x - 2j)(x + 2j)$$

both in terms of its coefficients and roots.

After submitting this function denoted by the identifier FC by means of the former way the following statement will be formed:

```
FC /POLY/ 0, 20, 0, 5;
```

where the coefficients are arranged according to their order starting from the zeroth one.

In the latter case, the statement for the function denoted by the identifier FR takes the form

```
FR /ROOT/ 5, 0, [0, 2];
```

Note, that the statement begins with the multiplicative factor. The pair of complex conjugate roots is represented there by their real and imaginary part in square brackets. The sign of the imaginary part is arbitrary.

Each of these user functions can be then used with arbitrary arguments. For example, the equation $q_1 = 5v_1^3 + 20v_1$ can be written as $q_1 = FC(v_1)$, and the equation $q_2 = 5v_2^3 + 20v_2$ as $q_2 = FC(v_2)$.

4.5 Text of user-defined functions

The statement for impulse, tabular and polynomial functions has the form

$$function / type / list ;$$

function is a user-defined identifier of the function.

type is one of the function-type identifiers shown below and placed between slashes / /.

list is a list of parameters of the function separated by comas , . These parameters are defined for each of the functions in the preceding paragraphs.

TYPE	FUNCTION
PULSE	impulse function
TAB	function given by a table
POLY	polynomial given by coefficients
ROOT	polynomial given by roots

In function expressions, the user-defined function should be referred to by the statements

$$function(argument)$$

function is a user-defined identifier of the function.

argument is a numeric constant or a symbolic expression enclosed in parentheses ().

The function-definition statement must precede the statement in which the function is used. Note, that a user-defined function can be used in the same problem repeatedly with different arguments.

Chapter 5

Altered functions and events

Chapter sections

5.1	Insertion of altered functions	5-1
5.2	Transformed functions	5-2
5.3	Trimmed functions	5-3
5.4	Periodic functions	5-4
5.5	Text of altered functions	5-5
5.6	Events	5-5

Chapter overview. *Altered functions can be formed by transforming, trimming or repeating periodically standard functions (Chapter 3) or user-defined functions (Chapter 4). Events allow controlling equations or diagrams during their nonlinear analysis (Chapter 12).*

5.1 Insertion of altered functions

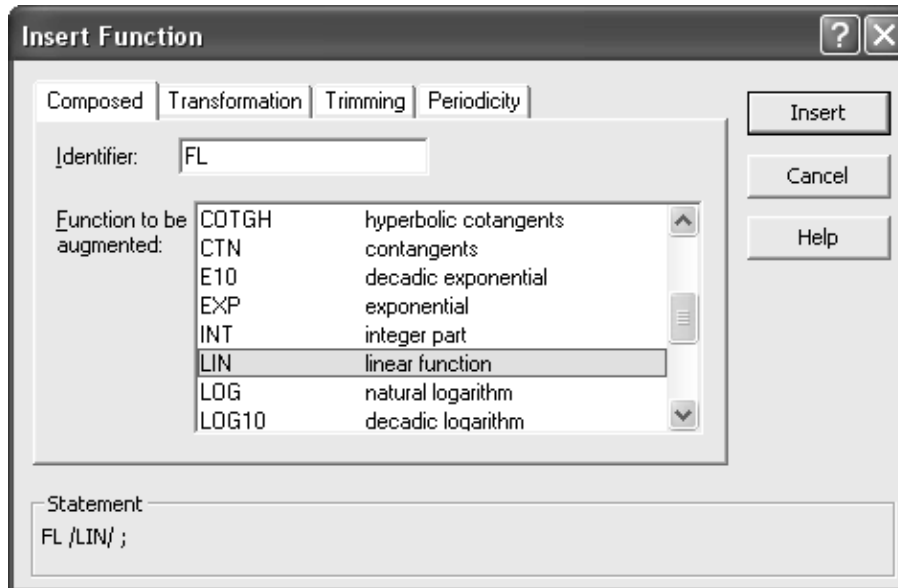
DYNAST forms an **altered function** $f(\bullet)$ by modifying a standard or user-defined function $g(\bullet)$ by transforming it, by trimming it outside an interval of its argument, or by repeating it periodically. The symbol \bullet stands for an arbitrary argument.

The dialog for forming Altered functions out of a standard function $g(\bullet)$ can be opened by selecting Inser Altered Function in the System menu. The standard-function alteration is done in the following steps:

1. Choose an identifier for the formed altered function $f(\bullet)$.
2. Select the standard function $g(\bullet)$ to be altered.
3. Select the way of alteration: Transformation, Trimming, or Periodicity.
4. Specify parameters for the alteration as explained in the next paragraphs.

You can repeat the Steps 3 and 4 to apply two or all three ways of alteration to the same function. Consider the impact of the order of these alterations.

User-defined functions can be altered using the tabs placed directly in the dialogs for inserting each of these functions.



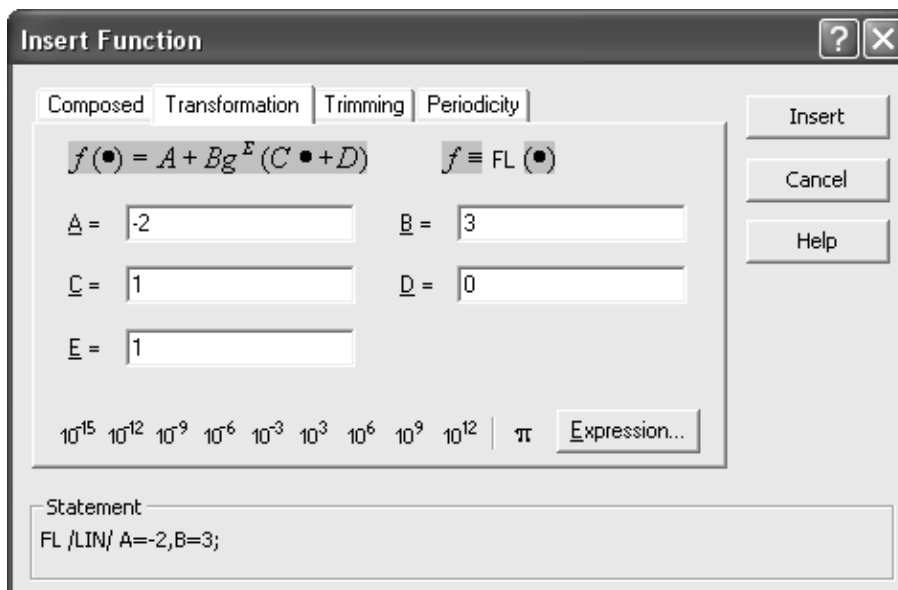
Selecting standard function to be altered.

5.2 Transformed functions

A function $g(\bullet)$ can be transformed into the following function

$$f(\bullet) = A + Bg^E(C\bullet + D)$$

using the tab Transformation in the dialogs. The values of the function parameters A , B , C , D , E can be specified by numeric constants or symbolic expressions. Their default values are $A = D = 0$, and $B = C = E = 1$.



Specification of function transformation.

Example. A function representing the linear relation $f(\bullet) = 3\bullet - 2$ denoted by the identifier `FL` can be formed by transforming the linear function `LIN` (identity) with parameters $A = -2$ and $B = 3$. The other parameter values remain default. DYNAST then generates the statement

FL /LIN/ A = -2, B = 3;

The equations $v = 3t - 2$ and $Q = 3p - 2$, for example, can be submitted as
 $v = \text{FL}(\text{time}); Q = \text{FL}(p);$

Example. The function $f(\bullet) = I_0(e^{\theta \bullet} - 1)$ denoted as D1 can be formed by transforming the standard function $\exp(\bullet)$ with the parameters $A = -I_0, B = I_0, C = \text{theta}$. This would result in the statement

D1 /EXP/ A = -I0, B = I0, C = theta;

This function can be then used, for example, in the equations $i1 = D1(v1); i2 = D1(v2);$

Example. The equation $q = -4 + 3[(2p - 1) + 5(2p - 1)^3]^2$ can be submitted as

F /POLY/ 0, 1, 0, 5, A = -4, B = 3, C = 2, D = -1, E = 2; q = F(p);

where the first statement was generated by DYNAST for a transformed user-defined polynomial function specified by its coefficients.

5.3 Trimmed functions

The screenshot shows the 'Insert Function' dialog box with the 'Trimming' tab selected. The 'Lower limit' L is set to 0, and the 'Upper limit' U is also set to 0. The 'Slope below the limit' SL is set to 10/1k, and the 'Slope above the limit' SU is set to 1k/30. The formulas for the slopes are displayed as $S_L = \left(\frac{df}{dx}\right)_{x < L}$ and $S_U = \left(\frac{df}{dx}\right)_{x > U}$. Below these, there is a row of scientific notation values from 10^{-15} to 10^{12} , followed by a π symbol and an 'Expression...' button. At the bottom, the 'Statement' field contains the code: D2 /LIN/ L=0,SL=10/1k,U=0,SU=1k/30;.

Specification
of function
trimming.

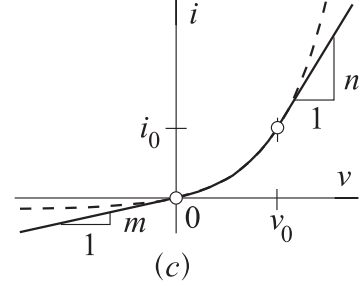
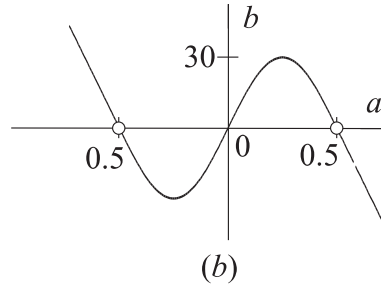
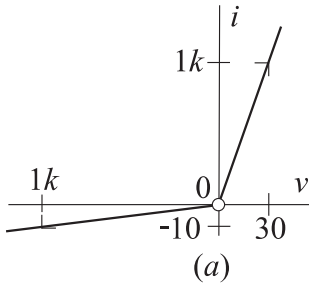
Using the Trimming tab in the dialogs a function $g(\bullet)$ can be trimmed to form the function $f(\bullet)$ linearly extrapolated outside the interval $L < x < U$ in such a way that

$$f(x) = \begin{cases} g(L) + S_L(\bullet - L) & \bullet < L \\ g(\bullet) & L \leq \bullet \leq U \\ g(U) + S_U(\bullet - U) & \text{for } \bullet > U \end{cases} \quad (5.1)$$

While L and U are limits of the trimming interval, S_L and S_U are slopes of the linear extrapolations outside this interval. Values of parameters L , U , S_L and S_U can be specified as numeric constants or symbolic expressions. The default values of L and U are $L = -\infty$ and $U = \infty$.

If the values for L and U are specified, but the values of slopes S_L and S_U are not specified, DYNAST chooses S_L and S_U in such a way, that the function $f(\bullet)$ is continuous at points $\bullet = L$ and $\bullet = U$, i.e.

$$S_L = \left(\frac{dg}{d\bullet} \right)_{\bullet=L} \quad \left(\frac{dg}{d\bullet} \right)_{\bullet=U}$$



Example. Using trimming, the piece-wise linear characteristic shown in Fig. a can be specified as

$$D2 \text{ /LIN/ } L=0, U=0, S_L=10/1k, S_U=1k/30; i = D2(v);$$

Example. Fig. b presents the exponential function $b = 30 \sin(2\pi a)$ trimmed within the interval $-0.5 < a < 0.5$ and transformed simultaneously as specified by the statement

$$FA \text{ /SIN/ } B = 30, C = 2\pi, L = -1/2, U = 1/2; b = FA(a);$$

Example. Fig. c shows the characteristic $i = I_0(e^{\theta v} - 1)$ linearized outside $0 < v < v_0$ in such a way, that its derivatives di/dv at both ends of this interval remain continuous, i.e., that

$$i = \begin{cases} mv & \text{for } v < 0 \\ I_0(e^{\theta v} - 1) & 0 \leq v \leq v_0 \\ n(v - v_0) + i_0 & v > v_0 \end{cases}$$

where $m = I_0\theta$, $n = I_0\theta e^{\theta v_0}$ and $i_0 = I_0(e^{\theta v_0} - 1)$.

This characteristic can be submitted as both transformed and trimmed exponential function

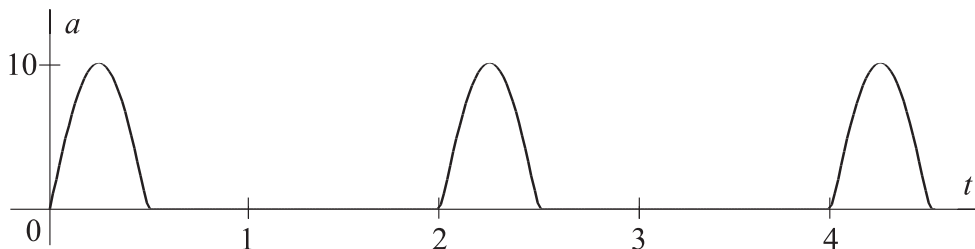
$$D3 \text{ /EXP/ } A = -I_0, B = I_0, C = \theta, L = 0, U = v_0; i = D3(v);$$

5.4 Periodic functions

Using the Periodicity tag, a function $g(\bullet)$ can be converted by DYNAST into the periodic function $f(\bullet)$ such that

$$f(\bullet) = g(\bullet) \text{ pro } 0 \leq \bullet < P \text{ a } f(\bullet + k \cdot P) = f(\bullet)$$

where k is an integer and P is the period of the function $f(\bullet)$.



Example. The plotted periodic function $a(t)$ can be formed by the simultaneous transformation and trimming of a sinusoid as

```
halfsin /SIN/ B = 10, C = 2pi, L = 0, U = .5, SL = 0, SU = 0, P = 2;
a = halfsin(time);
```

Example. A periodic impulse function can be easily formed from the impulse user-defined function, for example, as

```
Fimp /PULSE/ TD = -2, TR = 1, TT = 2, TF = 1, P = 5; v = Fimp(time);
```

However, the same periodic impulse function can be easily obtained by the following alteration of a tabular user-defined function:

```
Fimp /TAB/ 0,1, 1,1, 2,0, 3,0, 4,1, 5,1, P = 5; v = Fimp(time);
```

5.5 Text of altered functions

An altered function is specified by the statement

$$\textit{function} / \textit{type} / [\textit{list},] \textit{altlist};$$

function is a user-defined identifier of the altered function.

type indicates the type of the function to be altered placed between slashes / /. This can be a standard or a user-defined function.

list is a sequence of parameters of the altered user-defined function (if this is the case).

altlist is the sequence of altered-function parameters in arbitrary order separated by commas , .

5.6 Events

Insert Event

Name:

Order:

Expression:

10¹⁵ 10¹² 10⁹ 10⁶ 10³ 10⁰ 10⁻³ 10⁻⁶ 10⁻⁹ 10⁻¹² | π

Statement
EVENT STOP = time > 10;

Specification
of events.

Events represent changes in equation of diagram variables. The occurrence of a specific event is indicated by DYNAST during the numerical transient analysis by a change of the related event variable.

Until an event occurs, its **event variable** value is 10^{300} . When the event occurs, the event variable drops to the value of the `TIME` variable at the moment of the event occurrence. The **event order** is the integer number indicating the number-of-times the expression must become true for the event to occur.

Several events can be specified by a single statement of the form:

`EVENT event [(order)] = expression[, event [(order)] = expression...];`

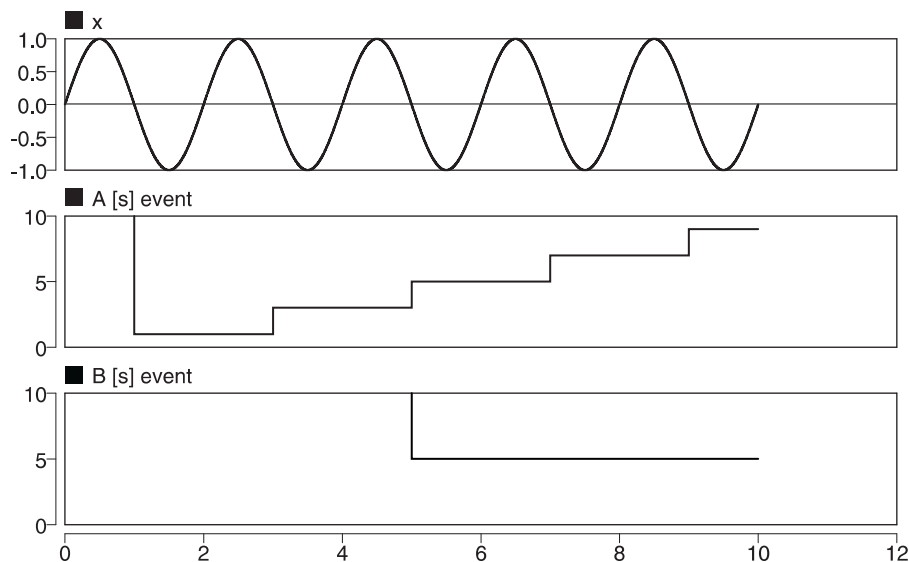
event is a user-defined identifier of an event as well as of the event variable associated with the event. If the identifier *event* is set to `STOP`, the DYNAST run is terminated at the moment of the event occurrence.

expression is a logical expression which defines the particular event. The event is assumed to occurs when *expression* becomes true.

order is a positive integer n indicating the event order.

- If the value of the order $n > 1$, the event variable changes its value from 10^{300} to the value of the `TIME` variable when its expression becomes true for the n -th time only, and then it keeps this value constant up to the end of the analysis
- In case *order* is set to '0', the corresponding event variable changes its value to a new `TIME` variable value each time its expression becomes true again.

The default values of the event order is 1.



Example. The figure shows waveforms of a variable $x(t)$ and of associated event variables the initial values 10^{300} of which are out of the figure range. The events are specified as

```
x = sin(1pi*time);
EVENT A (0) = x < 0; EVENT B (3) = x < 0; EVENT STOP = (time > 10);
```

Event A occurs each time, when $x(t)$ passes from positive to negative values. Event B occurs once only, when $x(t)$ passes from positive to negative values for the third time. The occurrence of event `STOP` results in termination of the analysis.

Chapter 6

Nonlinear equations

Chapter sections

6.1	Equations and their variables	6-1
6.2	Submitting explicit equations	6-2
6.3	Submitting implicit equations	6-3
6.4	Text of nonlinear equations	6-6

Chapter overview. *You will learn here how to submit systems of nonlinear algebraic, differential, or algebro-differential equations to DYNAST. The equations can be then evaluated or solved using the nonlinear analysis (Chapter 12). Linear algebro-differential equations can be solved also in a semisymbolic form (Chapter ??).*

6.1 Equations and their variables

When you are to compute the numerical value of a variable, you may encounter two different situations:

- the variable value can be obtained simply by **evaluating** a given expression
- the variable value can be obtained by **solving** an equation or a system of equations

We shall thus differentiate between evaluated variables and solved variables. Expressions for the evaluated variables are submitted to DYNAST in the form of explicit equations, whereas equations to be solved are submitted in an implicit form.

Example. If you want to obtain the unknown variables x specified by the quadratic equation

$$x^2 + bx + c = 0 \tag{6.1}$$

with the parameters b and c given, you must *solve* this *implicit-form* equation.

If you, however, recollect the formula

$$x_{1,2} = -b/2 \pm \sqrt{(b/2)^2 - c} \tag{6.2}$$

you can obtain x_1 and x_2 just by *evaluating* it using this *explicit-form* equation.

6.2 Submitting explicit equations

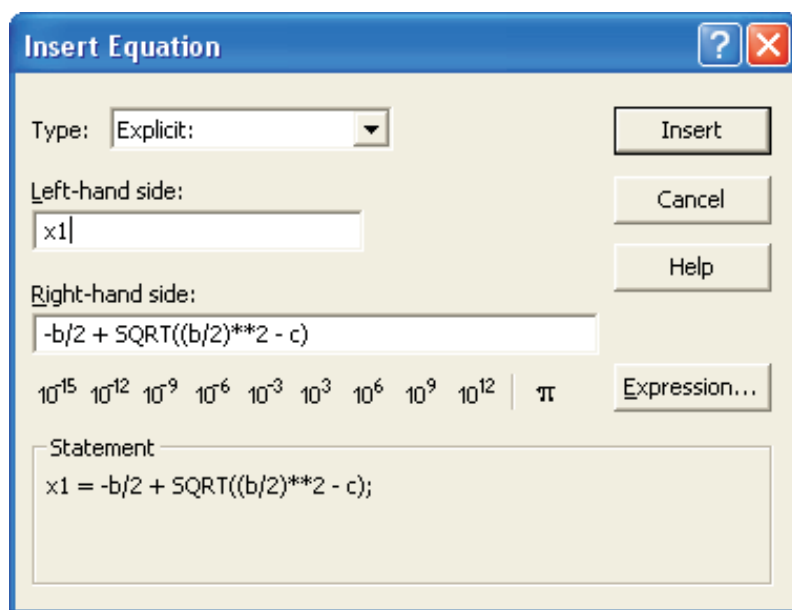
In general, a nonlinear **explicit-form equation** submitted to DYNAST to obtain the **evaluated variable** $y(t)$ is assumed to be in the form

$$y(t) = g(z_1, z_2, \dots, z_n, \dot{z}_1, \dot{z}_2, \dots, t) \quad (6.3)$$

where $g(\cdot)$ is a known function, and t is the independent variable (TIME). Numerical values of variables or parameters $z_1(t), z_2(t), \dots$ and of their derivatives $\dot{z}_1, \dot{z}_2, \dots$ must be submitted, evaluated or solved by DYNAST during its current execution before $y(t)$ is evaluated by (6.3).

To submit an explicit-form equation to DYNAST, follow these steps:

1. From the File menu, choose New and open a Problem text file.
2. From the System menu, choose Inser Equation to open the related dialogue.
3. Select Type: Explicit.
4. Enter an identifier for the evaluated variable into the Left-hand-side box.
5. Enter the rest of the equation into the Right-hand-side box. To do this, you can use the Expressions dialog as well as the suffix buttons (Chapter ??). The explicit equation is automatically copied into the bottom frame in the dialog so that you can easily check it.
6. Click Inser.



Submitting explicit equations.

Example. The dialog screenshot illustrates submitting the formula (6.2) for x_1 . Note, however, that DYNAST cannot evaluate (6.2) unless the numerical values of the parameters b and c are known. It is thus necessary to submit first two explicit-form equations specifying the parameters, for example,

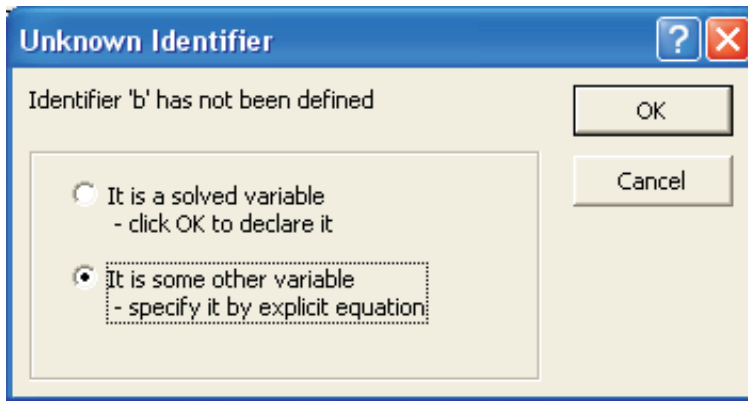
$$b = 3 \quad \text{and} \quad c = -4$$

One equation needs to be evaluated only, of course, if it is submitted with all parameter values set numerically

$$x_1 = -3/2 + \sqrt{(3/2)^2 + 4}$$

You have is still another option for specifying parameter values before submitting an equation in its symbolic form. It gives you the dialog opened from the System menu by selecting Edit System Parameters. This option is especially useful for later modifications of parameter values.

If a user trying to evaluate (6.2) clicks the Insert button in the above dialog without specifying b and c first, DYNAST opens the Unknown Identifier dialog. It asks the user if b is a solved or evaluated variable. When the user selects the latter option, another Insert Equation dialog opens for specifying a numerical value of b . In a similar way, the user is forced to specify c . Then only inserting of (6.2) in its symbolic form into the problem text file is enabled.



Sorting
unspecified
variables or
parameters.

After submitting (6.2) to nonlinear static analysis (Chapter 12) to solve it for x_1 and x_2 , the following problem text file will be automatically generated by DYNAST:

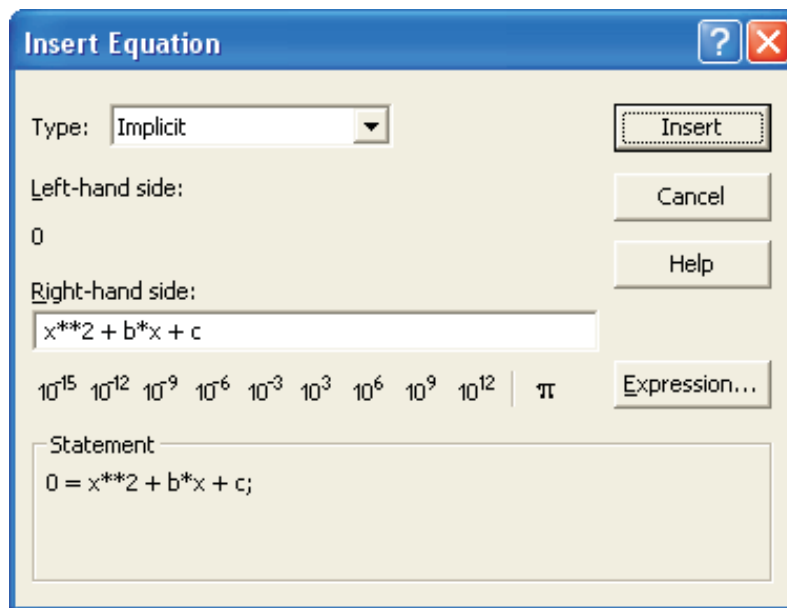
```
*SYSTEM;
b = 3; c = -4;
x1 = -b/2 + SQRT( (b/2)**2 - c);
x2 = -b/2 - SQRT( (b/2)**2 - c);
*TR; DC; PRINT x1, x2; RUN; *END;
```

6.3 Submitting implicit equations

A system of n algebraic, differential or algebro-differential **implicit-form equations** submitted to DYNAST for solving can be of the general implicit form

$$\begin{aligned} f_1(x_1, x_2, \dots, x_n, \dot{x}_1, \dot{x}_2, \dots, \dot{x}_n, z_1, z_2, \dots, t) &= 0 \\ f_2(x_1, x_2, \dots, x_n, \dot{x}_1, \dot{x}_2, \dots, \dot{x}_n, z_1, z_2, \dots, t) &= 0 \\ \dots &= 0 \\ f_n(x_1, x_2, \dots, x_n, \dot{x}_1, \dot{x}_2, \dots, \dot{x}_n, z_1, z_2, \dots, t) &= 0 \end{aligned} \quad (6.4)$$

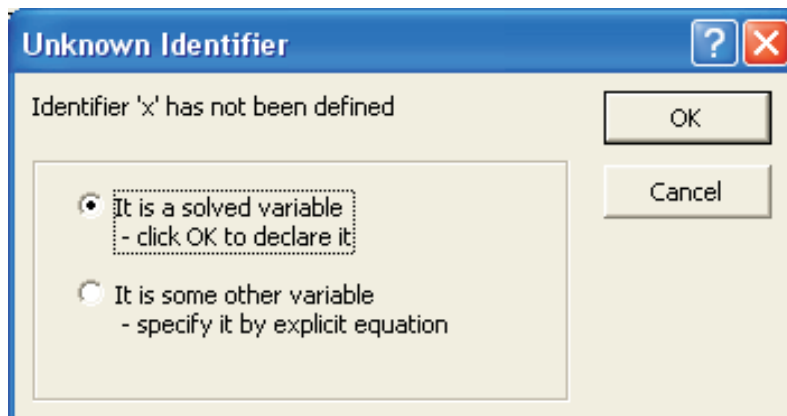
where $f_1(\bullet), f_2(\bullet), \dots, f_n(\bullet)$ are known functions, $x_1(t), x_2(t), \dots, x_n(t)$ are the **solved variables**, and $\dot{x}_1(t), \dot{x}_2(t), \dots, \dot{x}_n(t)$ are their derivatives with respect to the independent variable t (TIME). In the case of algebraic equations these derivatives are zero. The variables or parameters $z_1(t), z_2(t), \dots$ are assumed to be evaluated by DYNAST earlier during its current execution before solving the discussed equations. For submitting initial conditions see Chapter 12.



Submitting implicit equations.

The procedure for submitting implicit-form equations differs from that for submitting explicit-form equations in Step 4. In this case you should select Type: Implicit in the Insert Equation dialog. Characters $0 =$ will show up in the Left-hand-side box and you will be supposed to fill in the Right-hand side box. Note that in DYNAST notation for implicit-form equations their zero sides are always on the left.

Example. To solve the quadratic equation (6.1) fill in the dialog as shows its screenshot assuming that you have already specified values for parameters b and c . After clicking the Insert button the following dialog will show up. Then just click OK.



Confirmation of solved variable.

In this case DYNAST would generate automatically the problem text file

```
*SYSTEM;
b = 3; c = -4;
SYSVAR x;
0 = x**2 + b*x + c;
*TR; DC; PRINT x; RUN; *END;
```

During the data execution only one of the solutions x_1 and x_1 will be located. To find the other solution, an appropriate initial solution estimate must be specified (Chapter 12).

The operator of numerical differentiation d/dt of solved variables in the equations should be denoted by the characters `VD.` .

Example. After submitting the differential equation

$$\dot{x} = -2x + \sin 2\pi t$$

for solving and invoking its transient numerical analysis, DYNAST will generate the data

```
*SYSTEM;
SYSVAR x;
0 = -VD.x - 2*x + sin(2pi*time);
*TR; tr 0 4; PRINT x; RUN; *END;
```

To solve a differential equation of the ℓ -th order, the equation should be converted into ℓ equations of the first order using substitutions for the higher-order derivatives.

Example. The second-order Van der Pol nonlinear differential equation

$$\ddot{x} - \varepsilon(1 - x^2) + x = 0 \quad (6.5)$$

can be converted into two first-order differential equations using the substitution $\dot{x} = x_D$

$$\begin{aligned} \dot{x} - x_D &= 0 \\ \dot{x}_D + x - \varepsilon(1 - x_D^2)x_D &= 0 \end{aligned}$$

After submitting the explicit-form equation $\varepsilon = 0.01$ as well as the two implicit-form equations and invoking nonlinear transient analysis, DYNAST will generate the problem text file

```
*SYSTEM;
SYSVAR x, xD;
eps = 0.01;
0 = xD - VD.x;
0 = VD.xD + x - eps*(1 - xD**2)*xD;
*TR; TR 0 20;
PRINT x, xD; INIT xD = 1; RUN; *END;
```

Example. Similarly, the second-order time-dependent Bessel's differential equation

$$t^2\ddot{y} + t\dot{y} + (t^2 - k^2)y = 0 \quad (6.6)$$

can be converted into two first-order equations

$$\begin{aligned} y_D - \dot{y} &= 0 \\ t^2\dot{y}_D + ty_D + (t^2 - n^2)y &= 0 \end{aligned}$$

Then, after submitting the equations for numerical transient analysis, assuming that $n = 2$ has been submitted as an explicit-form equation, DYNAST will generate the problem text file

```
*SYSTEM; n = 2;
SYSVAR y, yD;
0 = yD - VD.y;
0 = time**2*VD.yD + time*yD + (time**2 - n**2)*y;
*TR; TR 0 10; INIT yD=.5; PRINT y; RUN;
```

6.4 Text of nonlinear equations

The data for submitting equations to be solved by DYNAST should be placed in the SYSTEM section of the problem text file.

Explicit-form equations can be submitted to DYNAST using the statement

$$evaluated = expression ;$$

evaluated is a user-defined identifier of the evaluated variable or parameter

expression is a numeric constant or symbolic expression.

Implicit-form equations should be submitted using the statement

$$0 = expression;$$

$0 =$ is the string introducing the implicit-equation statement

expression is a symbolic expression specifying the relation set to zero by the implicit equation.

The specification of a system of implicit-form equations must be preceded by a declaration list of all the system solved variables by the statement

$$\text{SYSVAR } solved [, solved...];$$

solved is a user-defined identifier of a solved variable

Notes:

- the independent variable t is denoted in the expressions by the identifier `TIME`
- A numerical value of any parameter or variable in each equation expression must be evaluated numerically before it used there as an argument. The solved variables and the independent variable denoted as `TIME` make the only exceptions.
- The operator $d \bullet / dt$ of numerical differentiation is denoted in the expressions by the identifier `VD. •`, where \bullet is related to a solved variable.

Chapter 7

Physical diagrams

Chapter sections

7.1	Principles of multipole modeling	7-1
7.2	Variables of multipole models	7-3
7.3	Automated formulation of equations	7-5

Chapter overview. *Whereas the block diagrams represent systems of equations, physical diagrams portray configurations of real dynamic systems and energy interactions between the subsystems. A physical diagram can be set up based on mere inspection of the modeled real system without formulating any equations. For physical diagram set up using its schematic editor, DYNAST is able to formulate the necessary equations automatically. This chapter outlines the principles utilized in DYNAST for this purpose.*

7.1 Principles of multipole modeling

7.1.1 Approximating assumptions

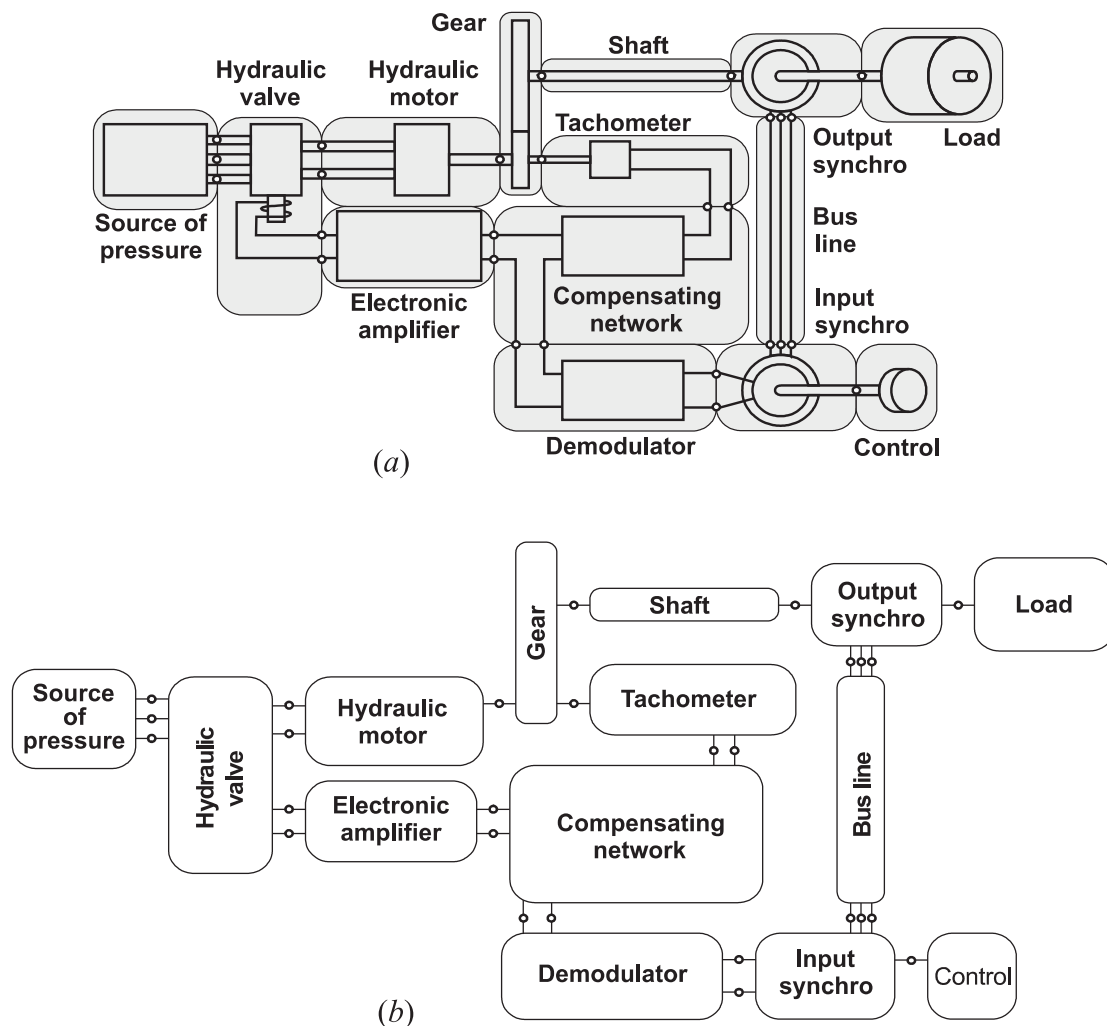
Behavior of a real dynamic system is related to the flow of energy and matter between the system and its surroundings as well as between the subsystems of the system. Energy and matter may change their forms inside the subsystems, or they can be accumulated there and released later. Energy can be also dissipated in the subsystems in the form of heat.

To investigate the flow of energy ¹ between subsystems rigorously, we should enclose each subsystem into an imaginary closed surface – a **subsystem interaction boundary** – and to integrate infinitesimal flows all over the boundary areas. The interaction boundaries must be chosen in such a way that neither energy nor matter may accumulate or change its form in the space between the boundaries.

In most of practical measurements or calculations of energy or matter flows between subsystems we resort tacitly to the following approximations:

¹In this text we will consider the energy flow only as the flow of matter $\dot{m} = dm/dt$ is usually converted into the volume fluid flow $Q = \dot{m}/\rho$, where ρ is the mean value of the specific mass of the passing matter.

- The flow of energy or matter to or from each subsystem takes place in a limited number of **energy inlets**, i.e. regions in the interaction boundary. The inlet regions represent cross-sections of electrical conductors, pipes with fluid, shafts, etc., passing through the boundary.
- Flow of energy or matter through each such region can be approximated by the product of a pair of **power variables**(see Table 7.1).



Example. Fig. *a* shows an example of a real system exploiting physical phenomena from several energy domains (mechanical, electrical and fluid). The individual subsystems are separated there by interaction boundaries. These surfaces share with each other the regions of mutual interactions denoted by small circles.

7.1.2 Multipole models of dynamic systems

Multipole modeling is based on the above mentioned approximating assumptions. **Multipoles** are models of individual real components characterizing their dynamic behavior. The interaction regions on a subsystem boundary are represented by **poles** of the subsystem multipole model. The number of poles of a multipole equals to the number of power-variable pairs necessary for the approximation of subsystem energy interactions.

Table 7.1: Pairs of through- and across-variables

Physical domain	Power variables		Energy variables	
	through i	across v	through $\int i dt$	across $\int v dt$
electrical	Electrical current [A]	electrical voltage [V]	electrical charge [C]	flux linkage [V.s]
magnetic	magnetic flow rate [Wb/s]	magnetic voltage [A]	magnetic flow [Wb]	
fluid or acoustic	volume flow [m ³ /s]	pressure [N/m ²]	volume [m ³]	pressure momentum [N.s/m ²]
translatory	force [N]	velocity [m/s]	momentum [N.s]	displacement [m]
rotary	torque [N.m]	angular velocity [rad/s]	angular momentum [N.m.s]	angular displacement [rad]

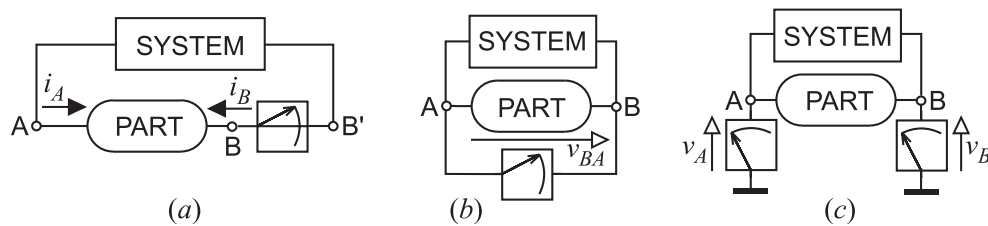
Multipoles can be represented graphically by symbols with poles pictured by short line segments sticking out of the symbols and called **pins**. The multipole model of a complete system can be represented by a **physical diagram**. The diagram consists of the multipole symbols and of nodes interconnected by lines with some of the multipole pins. Each **node** represents a place of mutual interaction between particular subsystem energy inlets represented by the pins. The lines interconnecting multipole pins can be considered as ideal interconnections transferring energy or matter without its change or accumulation.

Example. Fig. *b* presents a physical diagram of the system shown in Fig. *a*. The multipoles modeling individual subsystems are denoted there by symbols in the form of closed curves corresponding to the subsystem interaction boundaries. Small circles used for denoting energy interactions in Fig. *a* denote nodes in Fig. *b*.

7.2 Variables of multipole models

7.2.1 Power and energy variables

One of the variables in each pair of power variables is a **through variable** while the other one is an **across variable**. Table 7.1 shows also examples of **energy variable** pairs, i.e. the variables which can be evaluated by integration of power variables over time. The empty window in the table corresponds to the variable the existence of which has not been observed.



The through and across variables differ in the way in which they are measured (directly). Let us assume we want to measure the through variable i_B that enters a subsystem through its inlet B as shown in Fig. *a*. Then we must disconnect the inlet B from the rest of the system and re-connect it back via a measuring instrument. Measurement an across variable v_{BA} shown in Fig. *b* does not require subsystem disconnection. The measuring instrument is connected between distinct places of the system, i.e. between the inlets A and B in this case.

One of the nodes is always considered as a **reference node**. Across variables of non-reference nodes with respect to the reference node, like v_A and v_B in Fig. *c*, are **node across variables**. The across variables of poles with respect to the reference node are **pole absolute across variables**. An across variable between non-reference nodes or between poles is the **relative across variable**.

Table 7.2: Across-variable references

ENERGY DOMAIN	ACROSS VARIABLE	SYMBOL	EXAMPLE
electrical	electrical voltage		electrical ground
magnetic	magnetic voltage		
fluid or acoustic	pressure		free atmosphere
mechanical	(angular) velocity		absolute frame

7.2.2 Variable orientation

The value of a variable is positive or negative depending on its orientation with respect to the assumed **polarity reference**. In this text, we shall stick to the following conventions with regard to polarity references.

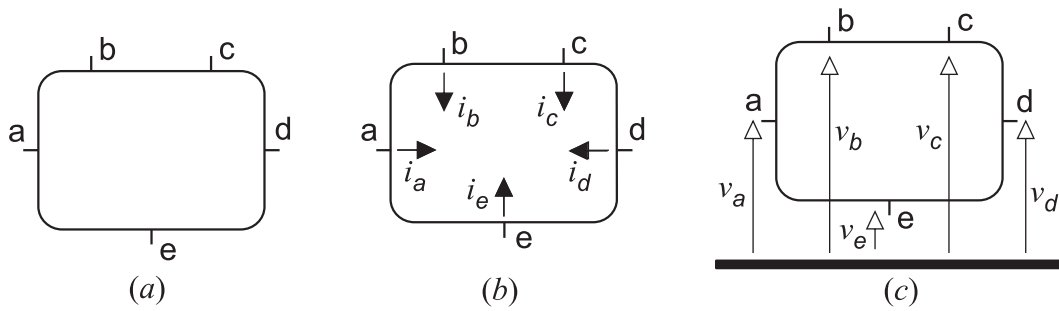
Orientation of across variables:

- Polarity of across variables is denoted by arrows with empty arrowheads.
- Each such arrow situated between two places will always will point towards the place of the assumed larger value of the absolute across variable.

Orientation of through variables:

- Polarity of through variables is denoted by arrows with full arrowheads.
- Each such arrow will point in the assumed positive direction of motion of the related medium.

For example, the medium may consist of mass particles or bodies in mechanical domain, or of positively charged particles in the electrical domain.



Example. Fig. *a* shows an example of a fivepole which is used to illustrate the assumed positive orientation of pole through variables in Fig. *b*, and of absolute pole across variables in Fig. *c*.

The total **power input** of an n -pole is defined as

$$P(t) = \sum_{j=1}^n i_j(t) \cdot v_j(t) \quad (7.1)$$

where i_j is the through variable and v_j is the absolute across variable of the j -th pole. Therefore, we assume that the multipole power input is positive when energy is consumed (i.e. dissipated or accumulated) in it. If energy is supplied from a multipole into the system, the power input of the multipole is negative.

7.3 Automated formulation of equations

7.3.1 Postulate of continuity and compatibility

The automated formulation of equations in DYNAST for a given physical diagram is based on the postulates of continuity and compatibility. Interpretation of these postulates in the form of specific physical laws as they are known from some energetic domains are given in Table 7.3. The postulates cannot be proven, but no contradicting phenomena were encountered in the field of the classic non-relativistic physics used for solving most of the engineering problem.

Table 7.3: Postulates of continuity and compatibility.

DOMAIN	POSTULATE OF CONTINUITY	POSTULATE OF COMPATIBILITY
electrical	Kirchhoff's current law	Kirchhoff's voltage law
magnetic	continuity of magnetic flux	Ampere's circuital law
fluid	principle of mass conservation	principle of pressure composition
mechanical	dynamic equilibrium of forces	principle of motion composition

According to the postulate of continuity

$$\sum_k i_k = 0 \quad (7.2)$$

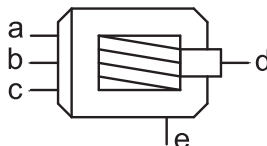
where $i_k, k = 1, 2, \dots$ are through variables entering a multipole. Obviously, this postulate is valid also for through variables entering a node of the multipole model.

According to the postulate of compatibility

$$v_{jk} = v_j - v_k \quad (7.3)$$

where v_j and v_k are absolute across variables of two poles of a multipole, and v_{jk} is the relative across variable between these poles.

In general, the postulates may not apply to the complete set of poles of a multipole. In any case, however, the poles of each multipole can be separated into disjoint subsets such that both postulates are valid for poles within each subset.



Example. In the case of the multipole modeling the three-phase induction motor shown below the postulates do not apply to the complete set of the poles as their variable come from different energy domains. However, they apply separately to the subset of ‘electrical’ poles a, b and c, and separately to the subset of ‘mechanical’ poles d and e. We can thus specify the postulate of continuity in this case as $i_a + i_b + i_c = 0$ and $i_d + i_e = 0$. According to the postulate of compatibility, for example, $v_{ba} = v_b - v_a$ or $v_{ed} = v_e - v_d$, but not $v_{da} = v_d - v_a$.



Postulates of continuity and compatibility apply also to ideal connections. Thus, in agreement with (7.3), we can write for absolute across variables v_a and v_b at the ends a and b of the connection shown above $v_a(t) = v_b(t)$, as $v_{ab} = 0$. Similarly, according to (7.2), the relation $i_a(t) + i_b(t) = 0$ can be written for through variables i_a and i_b .

In nonmechanical domains, the ideal connections represent ideal electrical conductors, ideal pipes with fluid, etc. In mechanical domains, an ideal connection can be imagined as a massless rigid rod of a fixed lengths. The rod transfers force from one end to the other without any change, and both ends of the rod have the same absolute velocity. The lengths of the rod, however, is not defined.

7.3.2 Constitutive relations of component models

As we mentioned already, the multipole way of modeling allows to separate specification of mutual energy interactions of system components from the specification of the behavior of individual components. It is assumed that each multipole model of a component can be characterized mathematically by **constitutive relations** between across and through variables of its poles.

While DYNAST is able to derive the relations (7.2) and (7.3) automatically for the physical diagram of a system, the choice of specific constitutive relations for the individual multipoles is up to the users. Fortunately, DYNAST component libraries contain large number of component multipole models – called **submodels** – with their constitutive relations specified already.

To choose an appropriate model for a component, you should take into consideration

- which of the component features you are interested in
- required accuracy of modeling
- magnitude of variable changes (see Table 7.4)
- time-rate of variable changes (see Table 7.5)
- component parameter changes over the modeling interval

Table 7.4: Models according to the magnitude of variable changes.

CHANGES	MODELS	EQUATIONS
small	linear	with constant coefficients
large	nonlinear	with variable-dependent coefficients
small and large	parametric	with variable- and time-dependent coefficients

A component-model development can be based, for example, on

- a hypothesis about the physical principles underlying the component behavior
- approximation of measured characteristics of the real component
- analysis of the model of internal configuration of the component
- reduction of an unnecessarily complex model of the component

In engineering practice some of these approaches are usually combined. For example, the model based on a physical hypothesis is refined with respect to measured data.

Table 7.5: Models according to the time-rate of variable changes.

CHANGES	MODELS	EQUATIONS
slow	static	algebraic
fast	dynamic with lumped parameters	ordinary differential
very fast	dynamic with distributed parameters	partial differential

DYNAST supports development of new component models in the following ways:

- a multipole model can be developed for each group of similar components just once, stored in a library and reused with modified parameters any time later
- this job can be done independently by specialists from different disciplines
- constitutive relations can be represented in a submodel by the diagram of the component internal configuration, by a set of equations, or by measured data, for example
- the model refinement or component replacement (e.g., replacement of electrical motor for a hydraulic one) can be done without interfering with the rest of the system model

The appropriate choice of suitable models requires a good physical judgement and experience. In any case, models should be as simple as possible. The more complex a model is, the more demanding is its development and identification. Experienced modelers usually start with simple models and improve them iteratively it appears necessary.

7.3.3 Multipole modeling step-by-step

Up to this point, we have focused our efforts on understanding the basics of multipole modeling. Armed with this knowledge, you are now in a position to tackle significant engineering problems. By using the following step-by-step systematic approach, you can solve a series of simple problems instead of one large, formidable problem.

Step 1: Sketch the system. Sketch the geometric configuration of the real system you want to model. The sketch should clearly indicate the system geometric features like dimensions, distances and positions (in a coordinate system) that are relevant for the system dynamics. Movable system parts should be drawn displaced arbitrarily from their reference positions in the positive directions with respect to the chosen coordinate system.

Step 2: Define the system. Identify an energy boundary of the system separating it clearly from the system surroundings and specify the external effects on the system.

Step 3: Decompose the system into subsystems. Identify energy boundaries of subsystems separating them from each other. The subsystems may represent actual components from which the real system is assembled, aggregates of such components, or just individual dynamic phenomena taking place within the real components or between the components and the system surroundings.

Step 4: Indicate the subsystem energy interactions. Indicate the energy inlets in the subsystem boundaries. Associate each of the inlets with a pair of power variables.

Step 5: Assign a multipole model to each subsystem. Look for a suitable submodel in the DYNAST libraries. If you cannot find one, create a new submodel exploiting one of the following approaches or combining them:

- set up a physical diagram for the submodel internal configuration using physical elements or submodels available in DYNAST
- formulate a set of algebro-differential equations expressing interrelations among the power variables of the submodel inlets
- characterize the interrelations by data measured on the real subsystem

Step 6: Form a physical diagram for the complete system. Using the DYNAST schematic editor, place submodel symbols in the diagram window. Interconnect the submodel pins with diagram nodes in conformity with the interactions of subsystem inlets. Denote the reference nodes. Specify the submodel external parameters.

You will find more detailed instructions in the next chapters.

Chapter 8

Physical elements

Chapter sections

8.1	Variety of physical elements	8-1
8.2	Variables of physical elements	8-6

Chapter overview. *Physical elements are very useful elementary twopoles. Each of them models a two-inlet component not necessarily linear, it can be nonlinear, time-variable or controlled by external variables or parameters. The physical elements form a multipurpose kit very useful for constructing submodels of complex components. They can be combined there with blocks and equations.*

8.1 Variety of physical elements

8.1.1 Elements dissipating or accumulating energy

Table 8.1 gives graphical symbols used in DYNAST diagrams for physical elements modeling dissipation or accumulation of energy. The entry for magnetic inductors is empty in the table as no related physical phenomenon is known. The notion of a mechanical resistor is not commonly used by the engineering community.

The physical elements are considered as **pure dynamic models** in the sense that each of them represents a specific dynamic effect characterized by a specific **constitutive relation** as shown in Table 8.1. The physical significance of the parameter p shown in the constitutive relations of different elements and their physical dimension are given in Table 8.2.

Pure conductors, resistors and dampers model dissipation of energy, i.e. the conversion of energy into heat. This energy cannot be retrieved and returned back into the systems. Pure capacitors and inertors model accumulation of energy by the virtue of an across variable. Thus an inductor represents accumulation of inertial energy due to either rectilinear or rotational motion of a mass. Pure inductors and springs model accumulation of energy by the virtue of a through variable. The energy accumulated in pure capacitors, inertors, inductors and springs can be retrieved back at any time.

In the case of the elements with an asymmetric symbol even the $+$ sign is omitted for simplicity. It is always assumed that the pin shown in Table 8.1 in the upper position is associated

Table 8.1: Physical elements dissipating or accumulating energy.

TYPE	G	R	C	L
NONMECHANICAL	CONDUCTOR	RESISTOR	CAPACITOR	INDUCTOR
electrical				
magnetic				
fluid or acoustic				
MECHANICAL	damper		inertor	spring
translational				
rotational				
CONSTITUTIVE RELATION	$i = p \cdot v$	$v = p \cdot i$	$i = p \frac{dv}{dt}$	$v = p \frac{di}{dt}$

Table 8.2: Parameter p of elements dissipating or accumulating energy.

Energy domain	Conductor or damper	Resistor	Capacitor or inertor	Inductor or spring
	$p = \frac{i}{v}$	$p = \frac{v}{i}$	$p = \frac{i}{dv/dt}$	$p = \frac{v}{di/dt}$
electrical	conductance [S]	resistance [Ω]	capacitance [F]	inductance [H]
magnetic	conductance [Ω]	resistance [S]	permeance [H]	
fluid or acoustic	conductance [m ³ /(Pa.s)]	resistance [Pa.s/m ³]	capacitance [m ³ /Pa]	inertance [Pa.s ² /m ³]
mechanical translational	damping [N.s/m]		mass [kg]	compliance [m/N]
mechanical rotational	torsional damping [N.m.s/rad]		moment of inertia [m ² kg/rad]	torsional compliance [rad/(N.m)]

with the + sign regardless if this sign is explicitly shown in the symbol or not.

Symbols for fluid or acoustic capacitors as well as rectilinear or rotational inertors have one pin only. To simplify the notation the pins corresponding to the -poles are omitted as they must be always associated with the system-model reference.

8.1.2 Sources of energy and related elements

Table 8.3: Sources of energy and ideal indicators.







TYPE	J	E
Generic	through-variable source 	across-variable source 
Mechanical translational	source of force 	source of velocity 
Mechanical rotational	source of torque 	source of angular velocity 
Constitutive relation	$i = p$	$v = p$

Table 8.4: Ideal indicators and sensors.





TYPE		
Generic	across-variable indicator 	through-variable indicator 
Constitutive relation	$i = 0$	$v = 0$

Table 8.5: Ideal switches and operational amplifiers.

TYPE	S	OA
Generic	ideal switch 	ideal operational amplifier 
Constitutive relation	if p is true then $v = 0$ if p is not true then $i = 0$	$p = \text{expression that is forced to zero by OA}$

Besides the energy storing and dissipating elements we need also pure physical elements modeling sources of energy. Such elements are usually used to model external energy reservoirs large enough relative to the amount of energy they supply to the system or absorb from it without undergoing any change either in their across or through variable. Two limiting cases are thus considered in which either an across variable or a through variable is specified independently

of the amount of energy drawn from or delivered into the source – a **source of across variable** and a **source of through variable**. DYNAST symbols and constitutive relations of sources are given in Table 8.3.

Parameter p of ideal sources of energy is a constant. Sources the parameters p of which are functions of time only are called **independent sources**, but not ideal.

Note that a source of zero across variable behaves as an **ideal closed connection**, while a source of zero through variable behaves as an **ideal open connection**. Therefore, a zero-valued source of an across variable can be utilized as an ideal indicator of its through variable. Similarly, the zero-valued source of a through variable acts as an ideal indicator of its across variable. DYNAST symbols for the *ideal indicators* that may be used for modeling real measuring instruments or sensors are shown also in Table 8.4.

Examples. Table 8.6 shows in its left column examples of simple dynamic systems from different energy domains. The first system – an RLC circuit – is supplied from an electronic source of electrical current. In the fluid system, a pump driven by a motor forces a constant flow of water from a reservoir into the open tank. Water is returned back to the reservoir through a long pipe with a restriction at its end. The rectilinear mechanical system is powered by the weight of a body suspended on a cable connected to a hydraulic damper by a helical spring via a pulley. The pulley friction is assumed negligible. The rotational mechanical system consists of a flywheel linked to a ventilator propeller and to an engine by a long shaft torsionally flexible.

The drawings of systems shown in the left column can be readily converted into physical diagrams shown in the right column of the table. The energy interactions in the examples are assumed to take place at the component interconnections denoted by the characters M and N. These interaction places are represented in the diagrams by their nodes. Each of the examples consist of four components. The component No. 1, modeled by a source, is considered as the system surroundings. Each of the system component numbered 2, 3 and 4 is modeled by a physical element representing its dominant dynamic effect. More realistic models could be set up for each component by combining several physical elements.

8.1.3 Nonlinear, time-variable and controlled elements

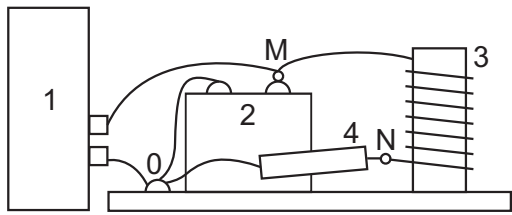
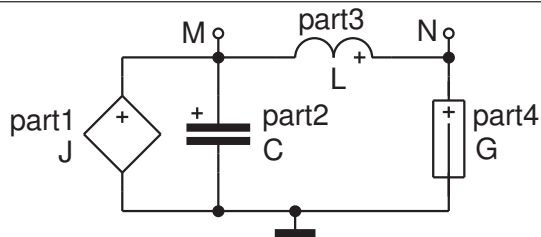
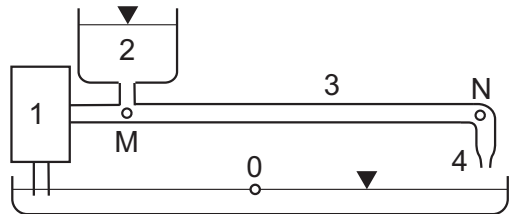
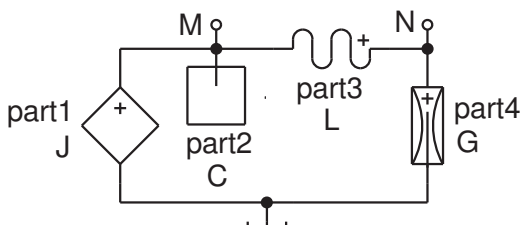
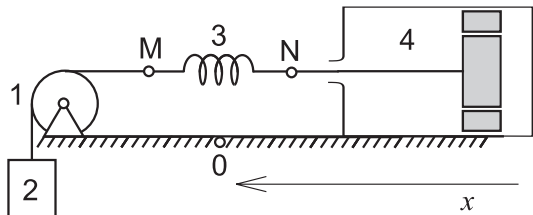
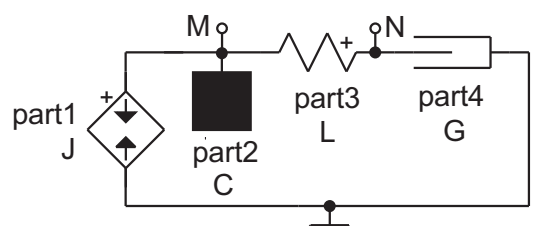
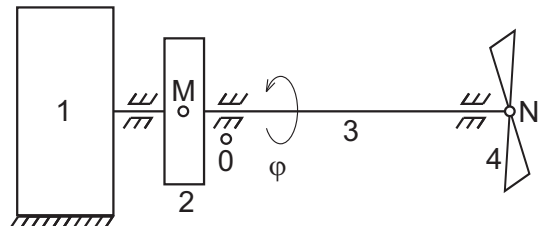
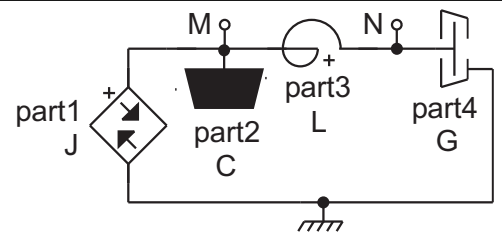
The parameter p of **ideal elements** is constant. The constitutive relations of physical elements submitted to DYNAST need not be constant, however. They can be nonlinear, time-variable or controlled by variables of some other elements, blocks, or equations in the system model. They can be also controlled by an ambient parameter like temperature, for example.

The parameter p of a physical element can be specified by a symbolic or tabular expression. In general, their constitutive relations can be of the form

$$p = f(z_1, z_2, \dots, \dot{z}_1, \dots, \dot{z}_1, \dots, t) \quad (8.1)$$

where z_1, z_2, \dots are variables or parameters of the system model and $\dot{z}_1, \dot{z}_2, \dots$ are their time derivatives.

Table 8.6: Examples of simple dynamic systems and their multipole models.

REAL SYSTEMS		SYSTEM MODELS
		
No.	Real components	Physical elements
1	electronic energy supply	source of electrical current
2	electrical condenser	electrical capacitor
3	coil of wire	electrical inductor
4	electrical resistor	electrical conductor
		
No.	Real components	Physical elements
1	pump driven by motor	source of volume flow-rate
2	open tank	fluid capacitor
3	long pipe	fluid inductor
4	fluid-flow restriction	fluid conductor
		
No.	Real components	Physical elements
1	weight of suspended body	source of gravitational force
2	suspended body	inertor
3	helical spring	spring
4	oil-filled damper	damper
		
No.	Real components	Physical elements
1	combustion engine	source of torque
2	flywheel	rotational inertor
3	long flexible shaft	torsional spring
4	ventilator propeller	rotational damper

8.1.4 Forbidden element configurations

Independent sources can be used in a very flexible way, yet not without any restrictions. Some of their configurations may breach the modeling principles due to an over-idealization of real systems. Fig. 8.1a shows a system model with three independent across-variable sources forming a close loop. If the across variables of the sources do not satisfy the postulate of compatibility the system equations have no solution. If, on the other hand, this postulate is satisfied, the system equations are singular unless one of the sources is removed from the loop (removing the source will not influence the system variables).

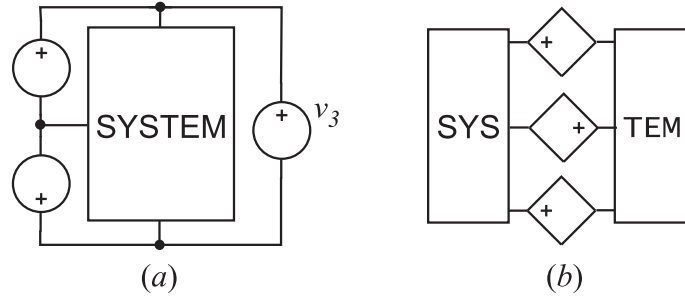


Figure 8.1: (a) Loop of across-variable sources, and (b) cut-set of through-variable sources.

The three through-variable sources shown in Fig. 8.1b form a cut-set, i.e. a set of sources, which – when removed from the system model – decompose the model into separated parts. If the through variables of the sources do not satisfy the postulate of continuity the system equations will have no solution. To prevent the equation singularity in case the postulate is satisfied, one of the sources must be removed.

Note, that any physical element with the zero value of its parameter behaves as a source.

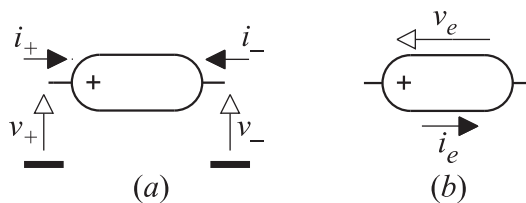
8.2 Variables of physical elements

8.2.1 Power consumption of physical elements

Physical elements are two-pole dynamic models of real system components based on the approximating assumption that the component power consumption can be expressed as

$$P(t) = i_e(t) \cdot v_e(t), \quad (8.2)$$

where $i_e(t)$ is the **element through variable** and $v_e(t)$ is the **element across variable**. Examples of the related physical quantities from several energy domains were given in Table 7.1.



The figures show a diagrammatic symbol of a generic domain-independent physical element. The line segments sticking out of the oval – the **pins** – represent **poles** of the physical element. The **element polarity** is indicated by the + sign in the vicinity of one of the poles.

This is the element +pole, while the other one is the element -pole (though the - sign is not shown in the symbol for simplicity). Absolute across variables v_+ and v_- are the **pole across variables**, i_+ and i_- are the **pole through variables**. Arrows indicate the assumed positive orientation of these variables.

8.2.2 Orientation of element variables

The sign determination of physical elements can be simplified by introducing the **element across variable** v_e and the **element through variable** i_e . We shall assume that the element variables are related to the pole variables of each element by the following convention:

$$v_e = v_+ - v_- \quad i_e = i_+ = -i_-$$

The previous figure illustrates the following convention: the arrow for the polarity reference of the element across variables will be always directed from the element -pole towards its +pole. The full-head arrow for the polarity reference of the element through variables will be always directed in the opposite direction. The impact of these conventions is, that we do not need to use the arrows for oriented elements at all.

Note that if the element across and through variables are both positive or both negative the power (8.2) is consumed in the element. If the signs of these variables are different, the element delivers power into the system. We can thus refer to this assumption as the **consumed energy convention**.

Variables of non-mechanical elements

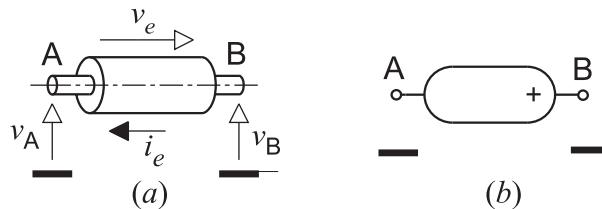


Fig. *a* shows a real component the energy interactions of which via its two energy inlets A and B are of a non-mechanical nature. The inlets may represent electrical terminals, fluid inlets, etc. The real component is supposed to be modeled by physical element shown in Fig. *b*. Orientation of the element with respect to the component was chosen by chance in such a way that the element +pole represents the component inlet B, and the -pole the inlet A. This interrelationship implies unambiguously, that the element across variable v_e corresponds to the component across variable v_{BA} , i.e. the across variable measured at the inlet B with respect to the inlet A. It is obvious at the same time, that the element through variable i_e corresponds to the flow of the component medium from the B inlet towards the inlet A.

Variables of mechanical elements

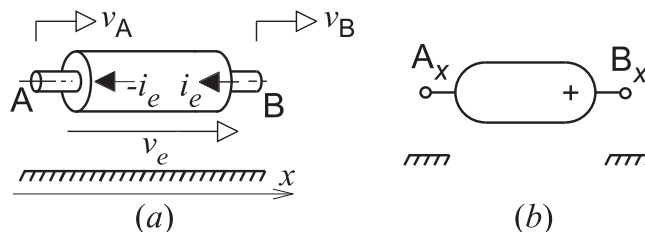
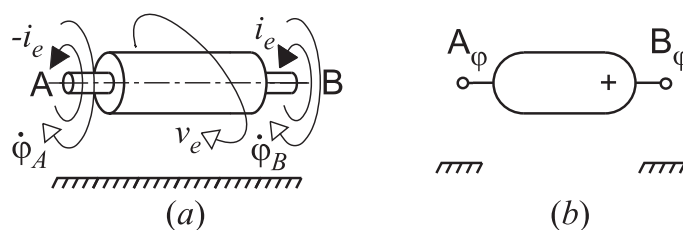


Fig. *a* illustrates a mechanical component the energy inlets A and B of which are in a translational rectilinear motion along the x -axis. Once the direction of the x -axis is given it is advantageous to orient a physical element modeling the real component in the following way. The value of the element across variable v_e should be positive if the distance between the inlets A and B increases while v_e should be negative if this distance decreases.

The figure also indicates the unambiguous interrelationship between the element through variable i_e and forces acting in the component if the element orientation is chosen in the recommended way. The element through variable i_e represents the reaction force acting at the component inlet corresponding to the element +pole to counterbalance the external force stretching the components. If the component is compressed, the sign of the related i_e becomes negative.



A similar reasoning can be applied to modeling a component with two energy inlets in a rotational motion about a fixed point shown in this figure. First it is necessary to choose the positive direction of rotation common to both inlets. It is helpful again if the positive value of the element across variable v_e corresponds to the case when the angle between the component inlets increases. The element through variable i_e corresponds to the reaction torque acting at the component inlet related to the element +pole to counterbalance the external torque.

Chapter 9

Block diagrams

Chapter sections

9.1	Block diagram architecture	9-1
9.2	Basic blocks	9-2
9.3	Block diagrams and physical models	9-5

Block diagrams are graphical representations of systems of equations. This chapter will introduce you to the very flexible way of block-diagram implementation of equations in DYNAST. Also equations for physical models of dynamic systems are considered. As DYNAST solves all the equations underlying block diagrams simultaneously, it is very robust and has no ‘algebraic loops’ problem. Blocks can be freely combined with multipoles and equations.

9.1 Block diagram architecture

Individual blocks are denoted in a block diagram by specific graphical symbols. Each block symbol represents a specific mathematical relation between the input and output variables of the block. Propagation of a variable from a block output to a block input is usually indicated by a line interconnecting the related output and input pins. Pins are short line segments sticking out of the block symbols.

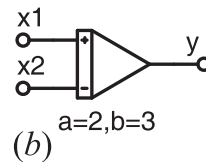
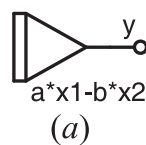
Specific rules that apply to block diagrams submitted to DYNAST:

- Each output pin of a block is connected to a node of the diagram. The name of the output variable becomes identical with the name of the node.
- A node can be connected to any number of inputs the names of which become identical with the name of the node.
- An input pin should be connected just to one node (with the exception of basic blocks).
- The input and output pins of the same block must not be connected to a common node (with the exception of the basic implicit blocks).

There are two categories of blocks in the diagrams submitted to DYNAST: basic blocks and submodel blocks. Differences between these two categories outlines the following table.

Table 9.1: Properties of blocks and their symbols.

BASIC BLOCKS	SUBMODEL BLOCKS
Block relation is defined inside DYNAST.	Block relation is defined in an external file.
Block symbol has just one output pin.	Block symbol has one output pin at least.
Block symbol has no input pins (with the exception of the transfer block).	Block symbol can have several input pins.
Each block instance is accompanied by the block relation in a textual form.	Only values of block parameters are added to an instance of a block symbol.



Example. The figures *a* and *b* show a basic and submodel block, respectively. Both block, however, represent subtracting integrators characterized by the identical relation

$$y = \int_{t_0}^t (ax_1 - bx_2) dt + y_0$$

Despite the fact that most of **basic blocks** have no input pin, the number of their input variables is not limited whatsoever. Any variable in the system model, even variables not associated with a node, can act as an input variable of a basic block. This feature provides an exceptional flexibility of block-diagram configurations. Variety of basic blocks with their relations is given in Table 9.2.

As the **submodel blocks** are specified in external files, their variety can be much larger than that of basic blocks. The DYNAST environment allows for easy extension of their variety including the graphical design of their symbols (Chapter 16). The number of input variables, however, is restricted by the number of block pins and each input variable must be associated with a system-model node.

9.2 Basic blocks

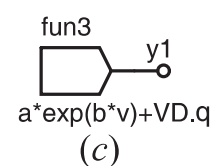
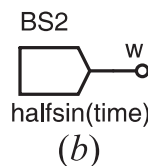
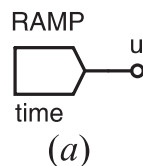
In the relations characterizing different basic blocks in Table 9.2 the input variables are denoted as $u_i(t)$, $i = 1, 2, \dots$ while the output variables as $y(t)$.

Explicit block

The output variable of an explicit block is defined by the explicit equation (6.3). The variables on the right-hand side of this equation represent input variables of the explicit block. If there is no such variable there (with the exception of time), such an independent explicit block acts as an autonomous **source** of its output variable.

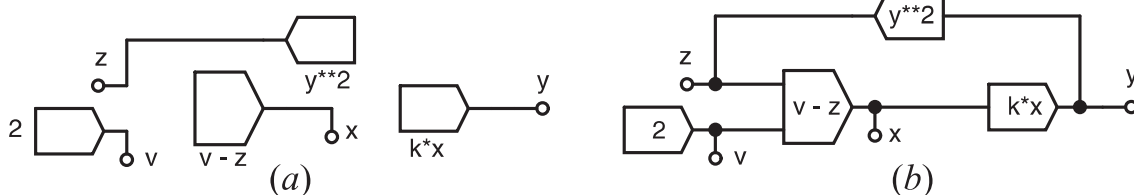
Table 9.2: Variety of basic blocks.

TYPE	BLOCK	SYMBOL	BLOCK RELATION
BS	explicit block		$y = f(u_1, u_2, \dots, \dot{u}_1, \dot{u}_2, \dots, t)$
BO	implicit block		$f(y, \dot{y}, u_1, u_2, \dots, \dot{u}_1, \dot{u}_2, \dots, t) = 0$
BI	integrator		$y = \int_{t_0}^t (k_1 u_1 + k_2 u_2 + \dots) dt + y_0$
BD	differentiator		$y = \frac{d}{dt} (k_1 u_1 + k_2 u_2 + \dots)$
BT	transfer block		$Y(s) = F(s) \cdot U(s), F(s) = K \frac{M(s)}{N(s)}$



Examples. The figure shows block symbols with their names placed above the symbols and the block relations below the symbols. The block in Fig. *a* can be used as the source of the ramp function $u(t) = t$. The block in Fig. *b* acts a source of periodic half-sinusoids $w(t)$, if *halfsin* is the altered function defined in Chapter 5. Assuming that $q(t)$ is a solved variable of the submitted problem, the block in Fig. *c* provides the relation

$$y_1(t) = ae^{bv} + dq/dt$$

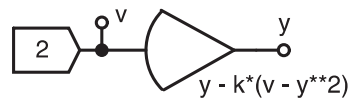


Example. The block diagram in Fig. *a* represents a simple static system with a quadratic feedback characterized by nonlinear algebraic equations. The system, excited at $t = 0$ by the step function of amplitude 2, consists from three explicit blocks (their names are hidden). Fig. *b* shows the same block diagram, only lines representing variable paths from nodes to block ‘inputs’ were added here for a better lucidity.

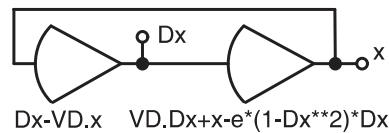
Implicit block

The implicit block represents one of the algebro-differential implicit equations from (6.4). The output variable of this block can be a function of not only other block output variables, but also of its own output variable. An input of this block can be thus ‘interconnected’ with its output (which is forbidden in the case of the other blocks). This feature allows for a wide applicability of implicit blocks.

Example. The feedback quadratic system from the previous example can be implemented by a single implicit block as



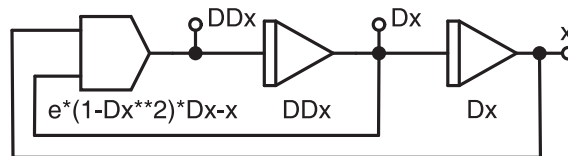
Example. The Van der Pol equation (6.5) can be submitted to DYNAST in the form of the block diagram



Integrator

The output variable of an integrator represents an explicit linear integral function of its input variables. The initial state of this variable is specified not in the diagram, but in the analysis section of the DYNAST input data (Chapters 12 and ??).

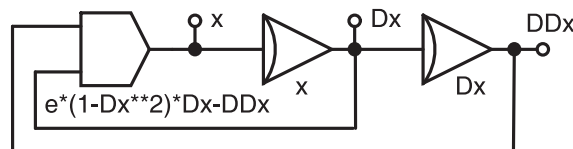
Example. The Van der Pol equation (6.5) can be also submitted as the block diagram with integrators



Differentiation block

A differentiation block represents an explicit first-order differential equation for its output variable.

Example. Another way of submitting the Van der Pol equation (6.5) shows the block diagram with 2 differentiation blocks



Transfer block

Transfer blocks are characterized by a rational-form transfer function $F(s)$, the argument of which is the Laplace transform variable s . Thus

$$F(s) = K \frac{M(s)}{N(s)}$$

where

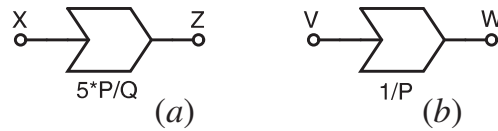
$$M(s) = s^m + a_{m-1}s^{m-1} + \dots + a_0 \quad \text{a} \quad N(s) = s^n + b_{n-1}s^{n-1} + \dots + b_0$$

K is a multiplicative constant.

Example. Transfer blocks with transfer functions

$$Z(s) = 5 \frac{s^2 + 1}{(s + 1 - j)(s + 1 + j)} X(s) \quad \text{and} \quad W(s) = \frac{1}{s^2 + 1} V(s)$$

are represented by symbols



assuming that the exploited polynomials have been submitted as

P /poly/ 1,0,1; Q /root/ 1, [-1,1];

9.3 Block diagrams and physical models

9.3.1 Block diagrams of physical models

Unlike physical diagrams, block diagrams do not represent configurations of real dynamic systems but systems of equations. Block symbols are interconnected by lines denoting a unidirectional propagation of a single mathematical variable or signal, not a bidirectional energy flow. The interconnections respect simple algebraic rules, but no physical laws.

The next example illustrates the paper-and-pencil procedure required before a block diagram representing dynamic of a physical system can be submitted to the computer.

Example. To set up a block diagram for analysis of the simple systems given in Table 8.6 using basic blocks we must first formulate the necessary equations following the next steps:

1. Constitutive relations for individual system components in the form of integral equations

Part	Physical element	Constitutive relation
1	source of through variable	$i_1 = J$
2	capacitor or inductor	$v_2 = 1/C \int i_2 \cdot dt$
3	inductor or spring	$i_3 = 1/L \int v_3 \cdot dt$
4	conductor or damper	$i_4 = G \cdot v_4$

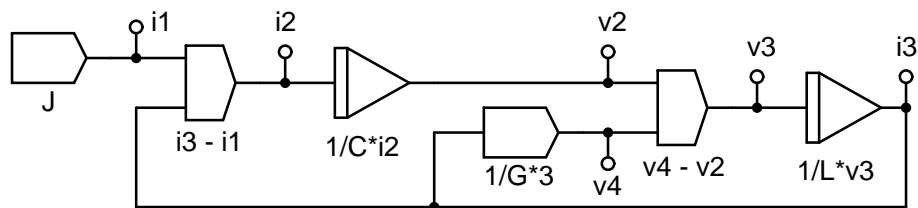
2. Algebraic equations for the postulates of continuity and compatibility

Postulate of continuity (7.2)	Postulate of compatibility (7.3)
$i_1 + i_2 - i_3 = 0$	$v_1 = v_2$
$-i_3 + i_4 = 0$	$v_3 = v_4 - v_2$

3. Rearranged non-singular explicit equations one for each variable

$$\begin{array}{ll}
 i_2 = J & v_1 = v_2 \\
 i_2 = -i_1 + i_3 & v_2 = 1/C \int i_2 \cdot dt \\
 i_3 = 1/L \int v_3 \cdot dt & v_3 = v_4 - v_2 \\
 i_4 = i_3 & v_4 = 1/G \cdot i_4
 \end{array}$$

4. Construction of the block diagram

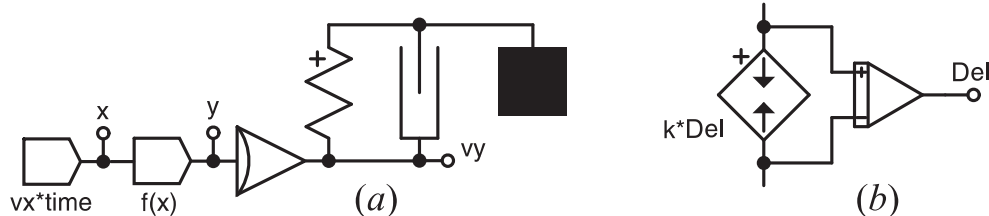


Interconnections between diagram nodes and basic-block ‘inputs’ are drawn here just for a better lucidity.

9.3.2 Blocks in physical diagrams

In problems submitted to DYNAST, blocks can be combined with multipoles and equations. Block behave as controlled sources of across variables the -pole of which is fixed to the system-model reference. Both the block input admittances and output impedances are zero. Therefore, block inputs do not ‘draw’ any energy from the system model, whereas block outputs can ‘deliver’ into the model any required amount of energy.

Example. The three physical elements shown in Fig. *a* form a simple model of a car-wheel suspension with the tyre following the curvy surface of a road pavement. The leftmost block acts as a source of the wheel position in the horizontal direction $x(t)$. The next block evaluates the function $y = f(x)$ describing the curvature of the road surface. The third block – the differentiator – acts here as a source of vertical velocity dy/dt required to excite the suspension model.



Example. Fig. *b* gives the length-controlled model of a linear spring where k is the spring stiffness and ℓ_0 is the spring length in the relaxed state. The spring model is formed by the source of force $F = k \cdot D\ell$ controlled by the spring prolongation $D\ell = \ell - \ell_0$ evaluated by the integrator block.

Chapter 10

Diagrams in graphical form

Chapter sections









10.1 Creating diagrams	10-1
10.2 Editing, printing and exporting diagrams	10-6

Chapter overview. *Using the DYNAST schematic editor you can easily create or edit diagrams in a graphical form. The diagrams are set up in a kit like way from physical elements, basic blocks and submodel symbols. They can be also combined with equations.*

10.1 Creating diagrams

10.1.1 Environment for creating diagrams

Table 10.1: Commands for graphical editing of diagrams.

ICON	COMMAND	DESCRIPTION
	Select	Selects object or sets neutral mode or
	Part	Opens libraries of part symbols
	Link	Draws link or multilink
	Link junction	Places junction of links or multilinks
	Multilink entry	Draws multilink entry
	Node label	Places non-reference node label
	Pole label	Places pole label in submodel diagram
	Text note	Places text in diagram

Diagrams can be created in a graphical form in the diagram window using the DYNAST schematic editor. To open the window for creating a new diagram

1. From the File menu, choose New.
2. In the New File dialog, select Dialog in the File type drop-down list.
3. Specify the File name and diagram Title.
4. Click the OK button.

Once the diagram window opens, diagram-related menus and a toolbar will appear at the top of the DYNAST main window and will stay there as long as the diagram window is active. The DYNAST commands for creating diagrams are available in the Place menu. Instead of these commands you may use the toolbar buttons given in Table 16.2.

Zooming diagrams. To zoom-in or zoom-out a diagram (together with the underlying grid) choose Zoom In or Zoom Out from the View menu.

Undoing or redoing actions. To undo or redo the most recent action you performed in a diagram choose Undo or – if you do not like the result – choose Redo from the Edit menu.

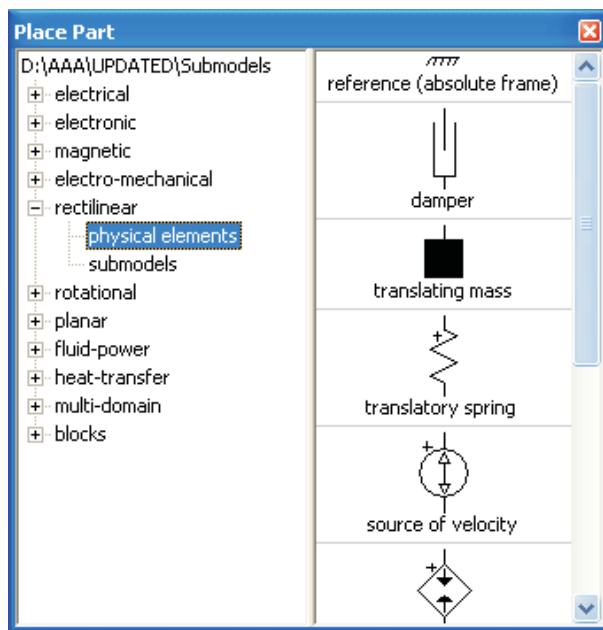
10.1.2 Placing parts

To place a part, i.e. the graphical symbol of a physical element, basic block or submodel into the diagram

1. From the Place menu, choose Part to open the Place Part dialog.
2. Select a part library and browse through the parts in it.
3. When you click the selected part and **RELEASE** the mouse button, the part symbol will become attached to the mouse pointer.
4. Move the part to the desired location in the diagram and click the location to place the part there.
5. You can repeat the last operation to place the part to several different locations in the diagram.
6. Right-click the mouse to terminate placing the selected part.

During the placement operation, when the part is attached to your mouse pointer, you can

- rotate it by pressing the R key
- mirror it horizontally by pressing the X key
- mirror it vertically by pressing the Y key





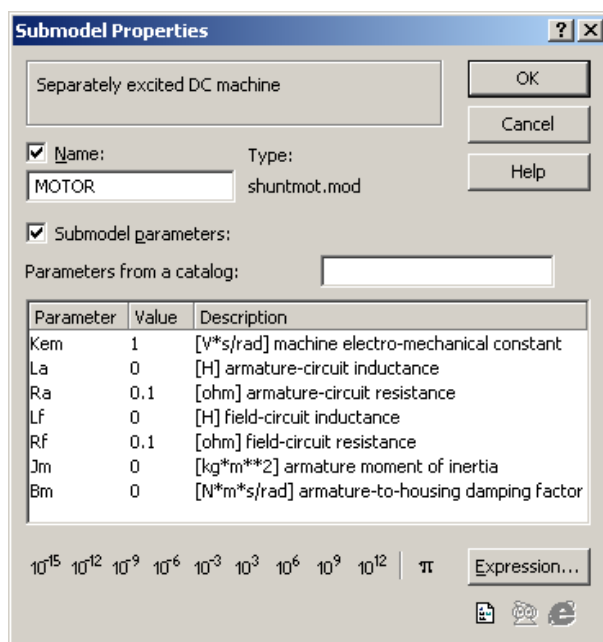
Selection
of part
from symbol
library.

The ends of pins of the placed part will get snapped to the grid underlying the diagram window. When you place a part in a diagram, you actually create an *instance* of the part. Each part instance should be given a specific name. If two part instances in a diagram have identical names an error is indicated by enclosing both part instances into yellow boxes.

Specification of parts. To specify properties of a part instance placed in the diagram

1. Double-click the part instance to open the part dialog.
2. Type in the instance Name and its Parameter value(s).
3. If you want to hide either the part name or parameter(s) in the diagram, clear the respective box in the part dialog.

For a detailed submodel description click the icon  or  in the bottom right corner.



Specification
of part
instance.

Placing text notes. To place a text note in a diagram

1. From the Place menu, choose Text note.
2. Move the Note symbol over the diagram to the desired location and click the location.
3. Right-click the mouse to stop placing notes.
4. Double-click the Note symbol to open its dialog and type in the text, then click OK.

10.1.3 Interconnecting parts

Parts have pins corresponding to signal or energy interactions of the components they represent. Multipins affiliate several pins. Pins have their names, and they are either generic or energy-domain specific (electrical, magnetic, thermal, fluid, and mechanical translational or rotational). To see the name of a pin, position the mouse pointer over it for a while.

Interconnecting pins. To interconnect some pins of part instances placed in the diagram

1. From the Place menu, choose Link.
2. Click in turn the tips of the pins to be interconnected by the same link (you may also click locations between the pins to deflect the link from a straight line).
3. Right-click the mouse to discontinue the link.
4. Go to the second step if you want to interconnect other some other pins.
5. Double-right-click the mouse if you do not want to interconnect any more pins.

When you connect a link to a pin, DYNAST provides a visual confirmation of the connection by removing the unconnected-pin box around the pin tip. If incompatible pins are connected (e.g., pins of different domains like electrical and mechanical, or a single pin with a multiple pin) an error is indicated by a yellow box in the diagram. Note please:

- If you place parts so that their pins meet end to end, the pins get connected.
- If you begin or end a link from the middle of another link, the links get connected, which is indicated by a thick dot.
- If two continuous links cross, they do not connect automatically.

Link crossings. To interconnect two crossed links in the diagram

1. From the Place menu choose Link junction
2. Click the mouse over the link crossing

Adjacent pins as well as pins interconnected by a link form a *node*. Each node in a diagram has a specific label. DYNAST labels nodes automatically by numbers after submitting the diagram for analysis. After placing the mouse pointer over the link connected to a node, the node number and domain appears in a short time.

10.1.4 Diagram nodes

Each group of interconnected pins as well as each unconnected pin form an independent **diagram node**. When an analysis is invoked for the first time for a new diagram, DYNAST denotes each node in the diagram by a number. You can see the number of a node when you position the mouse cursor for a while over a link to the node.

Labeling nodes. To give a your name to a non-reference node in the diagram

1. From the Place menu choose Node label
2. Move the label symbol over the diagram to such a location that the label pin touches the desired link and click the mouse
3. Repeat this operation or right-click the mouse to terminate it
4. Double-click each label to open its dialog and type in the node name

If two different nodes are given the same name, DYNAST indicates this as an error.

In each group of interconnected nodes one node at least must be the **reference node**. Otherwise the equations underlying the submitted diagram will be singular. A node becomes the reference node if a reference symbol from Table 7.2 is attached to it.

10.1.5 Using multilinks.

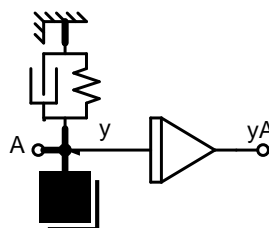
A **multilink** replaces several interconnecting links (like a busbar). To interconnect compatible multipins of part instances by multilinks proceed in the same way as in the case of links interconnecting single pins. If you place the mouse pointer over a multilink, its description displays in a short time.

To connect one link from a multilink to a single pin or node

1. From the Place menu, choose Multilink entry.
2. Click the mouse first over the multilink, and then over the tip of the target single pin or node.
3. Right-click the mouse if you do not want to connect the multilink entry to another pin or node.
4. Click the multilink entry to open the Attach Multilink Entry to Link dialog and type in the Link name.

If two continuous multilinks cross, they are not connected. To interconnect such multilinks proceed in the same way as in the case of single links. If the multilinks are compatible, their individual links will be interconnected properly.

Example. Diagram of a suspended body in the x - y plane with integrator evaluating y only.



10.1.6 Inserting equations in diagrams

Equations can be inserted into a diagram using the Insert Equation from the System menu. In a similar way, you can insert functions, events, etc. They will not project into the diagram, however. Read Chapter 11 to learn how to see input data in a textual form.

System parameters, i.e. parameters in the form of explicit equations can be inserted to diagrams or edited using the command Edit System Parameters from the System menu.

10.2 Editing, printing and exporting diagrams

Individual graphical objects (like part instances and their descriptions, links, multilinks, labels and text notes) or groups of these objects placed in the diagram can be rotated, copied, moved to another position, or deleted. First you must select the objects targeted for any such operation.

Selecting objects. To select a single object in a diagram, position the mouse pointer over the object and click it. The object will change its color. To deselect the object, click the mouse over a place outside the object.

A group of objects can be selected in two different ways:

- Hold the Ctrl key and click in turn each of the objects in the group.
- To select all objects in a rectangular area, click the left mouse button over one corner of the area, hold the button down and drag the mouse to the opposite corner, then release the button.

All objects selected in the group will change its color. To deselect the group of objects click the mouse over a place outside the group.

Rotating or mirroring objects. A selected object or group of objects can be

- rotated by pressing the R key
- mirrored horizontally by pressing the X key
- mirrored vertically by pressing the Y key

Moving objects. You can move a selected object or a group of objects to a new position in two different ways:

- Click the left button of the mouse over the selected object(s), hold the button, and drag the mouse to the desired position, then release the button.
- Hold down the spacebar while you press subsequently the arrow keys.

To move a link, select it and drag it to a new location; the link stretches to maintain its connectivity. If you want to move an object without maintaining its connectivity hold down the Alt key when dragging the object.

Copying objects. You can copy a selected object or a group of objects and to move it to a new position in two different ways similar to moving objects. Just hold down the Ctrl key during the operation. Remember, that you should give new names to the copied part instances or labels.

Deleting objects. To delete a selected object or a group of objects press the Del key.

Changing the diagram title. To change the title of a diagram, from the Edit menu choose Change Title to open the Diagram Title dialog.

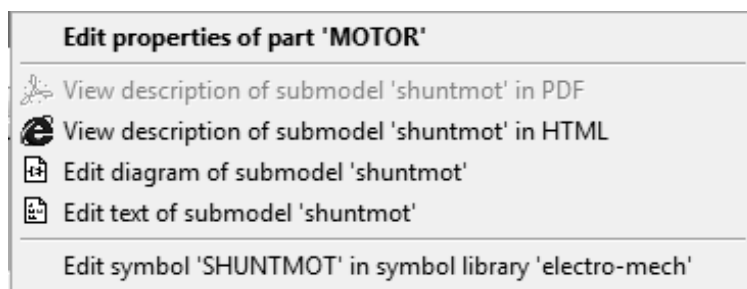
Placing text notes. To place text notes into a diagram

1. In the Place menu click Text note and release the button. The Note symbol will become attached to the mouse pointer.
2. Move the symbol Note to the desired location in the diagram and click the location to place the symbol there.
3. You can repeat the last operation to place the symbol to several different locations in the diagram.
4. Right-click the mouse to stop placing the symbol Note.
5. Double-click each of the placed Note symbols to open the dialog for inserting a text. Then close the dialog by clicking OK.

Placing frames. To improve lucidity of complex diagrams, you can enclose its sections into rectangular dashed frames. From the Place menu choose Rectangle and drag the mouse in the diagonal direction across the area of the desired frame.

10.2.1 Editing submodels from diagram window

Advanced users can edit submodel MOD text files, diagrams and graphical symbols directly from the active diagram window. After selecting the submodel symbol in the diagram and right-clicking it the following menu opens:



Example
of menu
for editing
submodel.

After editing a submodel using the tools in the menu and saving the result apply the command Refresh Libraries from the Edit menu.

10.2.2 Printing and exporting diagrams

To print an active diagram, choose Print or Print Preview from the File menu.

Exporting diagrams to postscript. To export the active diagram to postscript choose Export to Postscript from the File menu.

Exporting diagrams to clipboard. To export the active diagram as a bitmap to clipboard, choose Select All from the Edit menu, then click Copy.

Text files of diagrams. Diagrams are stored in ASCII text files with the extension DIA. Information necessary for the analysis of system models represented by diagrams is extracted by DYNAST from the diagram files and stored in problem files. These are ASCII text files with the extension PRB. Information is coded there in the DYNAST problem description language in the way specified in Chapter 11.

Chapter 11

Problems in textual form

Chapter sections

11.1 Diagrams and their textual files	11-1
11.2 Submitting texts of physical elements	11-4
11.3 Submitting texts of basic blocks	11-7
11.4 Submitting texts for submodel insertion	11-10
11.5 Catalogue of component parameters	11-12

Chapter overview. *Before submitting a diagram for number crunching, DYNAST converts it into a textual form. Some users prefer the textual to the graphical one, or they wish to work with both forms concurrently. Also those using DYNAST to solve equations submit their problems in the textual form. Such users can learn from this chapter how they can work with the textual data and how DYNAST supports this approach to problem submitting.*

11.1 Diagrams and their textual files

11.1.1 Synchronization of diagrams with problem text files

While graphical-form diagrams are stored in DIA files, their interpretation in the textual form DYNAST stores in PRB files. If you want to see the PRB file related to the diagram active on your screen, choose Problem or Submodel text from the View menu. To see the PRB file when the related diagram is not active, choose List of Problems from the View menu, or Open from the File menu.

Note that a diagram cannot be restored from the related PRB text file as this file contains information about properties and interconnections of diagram parts, but not about their geometric coordinates in the diagram.

Once you open a PRB file you can edit its text. If you are an advanced user you may wish to exploit the option of editing a problem both in the graphical and textual forms in turns. If this is the case, you will need both files synchronized with respect to each other after every change. Remember, that none of the files should contain a syntax error before the synchronization.

DYNAST provides the synchronization automatically. If you change a diagram and then open the related problem text file, DYNAST will propose you updates of the text file to achieve

synchronization with the diagram file. You will be similarly prompted to update the active diagram file with respect to the changed text file. If you accept the updates, DYNAST will provide them. If you will change your mind afterwards, you will still be able to cancel the updates by choosing the Undo command.

The following table shows the updates of diagram files proposed by DYNAST for different changes made in the related problem text files. For a changed diagram, DYNAST is capable of proposing synchronization updates of the corresponding problem text file without any exceptions.

CHANGE IN PROBLEM TEXT FILE	UPDATE OF DIAGRAM FILE
title changed	ignored with warning
part-parameter value changed	parameter value updated
part name or type changed	part removed
node name changed	ignored
equation added or changed	ignored
analysis added or changed	ignored

If you want to synchronize the related problem and diagram files regardless of the time when each of them was changed for last time, you can do this manually. From the Edit menu, choose Synchronize. Then select either Update diagram according to Text or Update text according to diagram.

11.1.2 Submitting diagrams in textual form

If you want to submit a problem in the textual form

1. From the File menu choose New to open the New File dialog.
2. Select Problem text in the File type drop-down list.
3. Specify the File name and the diagram Title.
4. Click the OK button to open the empty (almost) PRB file.

Textual specification of a diagrams is entered into the SYSTEM section of the PRB file. The configuration of the PRB files is as follows:

```

:: note
*: problem title
*SYSTEM;
parameter = constant; :: [physical unit] parameter description
parameter = constant; :: [physical unit] parameter description
...
physical elements, basic blocks, submodels, equations
...
analysis specification
RUN; :: title of analysis results *END;
```

```

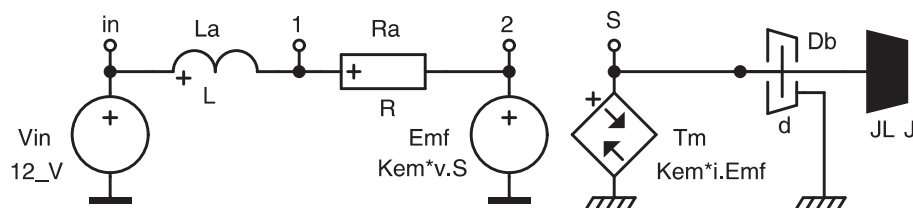
:: variable [physical unit] variable description
:: variable [physical unit] variable description
...

```

A very useful role is played in the PRB text files by commentary statements. All texts placed in a single line right of the semicolon character ; are considered as comments. The semicolon character can be also used for deactivating some statements during the problem debugging.

Comments beginning with :: play specific roles:

- When placed in the first line of a PRB file they introduce the text, that will be displayed as a note in the List of Problems opened from the View menu.
- In the SYSTEM data section they are used to clarify system parameters.
- Placed right of the RUN; statement, they are used as titles of tabular and plotted analysis results.
- Behind the statement *END; , they are used to clarify meaning of the computed variables in tables and plots.



Example. The following data illustrate the typical arrangement of a PRB text file without analysis specification. At the same time, the example demonstrates different ways of exploiting comments.

```

:: without analysis
*: DC motor start
*SYSTEM;
Kem=.05;    ::[V*s/rad] motor constant
L=1.5m;     ::[H] armature inductance
R=0.5;      ::[ohm] armature resistance
J=.25m;     ::[kg*m**2] moment of inertia
d=.1;       ::[N*m*s/rad] bearing resistance
:Electrical part
Vin > E in = 12_V;    :supply voltage
La in-1 = L;          :armature inductance
Ra 1-2 = R;           :armature resistance
Emf 2 = Kem*v.S;      :induced voltage
:Mechanical part
Tm > J S = Kem*i.Emf; :motor torque
Db > G S = d;          :bearing resistance
JL > C S = J;          :moment of inertia
*END;

```

```

::I.La      [A] motor electrical current
::Tm        [N*m] internal motor torque
::v.S        [rad/s] shaft angular velocity

```

11.2 Submitting texts of physical elements

11.2.1 Physical elements between nodes

For a survey of physical elements see Chapter 8.

If you choose Insert Element in the System menu, the following dialog opens:

The 'Insert Element' dialog box contains the following fields and controls:

- Type:** A dropdown menu showing 'R resistor'.
- Name:** A text box containing 'Rout'.
- In series:** An unchecked checkbox.
- Node+ :** A dropdown menu showing 'OUT'.
- Node- :** A dropdown menu showing '0'.
- Value:** A text box containing '5_kohm'.
- Expression...:** A button next to a row of scientific notation and pi symbols.
- Statement:** A text area containing the statement 'Rout OUT = 5_kohm;'.
- Buttons:** 'Insert', 'Cancel', and 'Help' buttons on the right side.

Submitting
texts of
physical
elements.

1. From the Type list choose the type of the submitted element in conformity with Chapter 8.
2. Type the name of the element in the Name text box.
3. Specify the name of the node to which the element +pole should be connected using the Node+ box.
4. In the Node- box specify the name of the node to which the element -pole should be connected.
5. In the Parameter box type the element parameter value as a numeric constant or symbolic expression.
6. Click the Insert button and add the required data if a dialog opens.

The following statement for physical element specification will be inserted to the location in the PRB file denoted by the cursor:

$$name \ [> type] \ node+ \ [- node-] \ [= parameter];$$

Notes:

- If *name* was chosen in such a way that its first one or two characters correspond to the element type identifier, the string *>type* can be omitted.
- If *-node* is 0 (i.e. if it is a reference node), the string *-0* can be omitted.
- If the parameter value is 1, the string *=1* can be omitted.

Example. All the elements in the previous example were specified as connected between nodes.

11.2.2 Physical elements in series

If a group of physical elements of type R, L, E, or S forms a series configuration, the group can be submitted in a simplified way. This saves memory and speeds up the computation as series configurations are characterized in DYNAST by one equation only. It is not possible, however, to evaluate across variables of the individual elements in the series configuration.

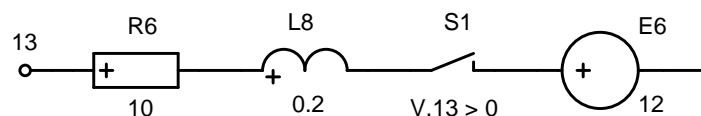
To submit a series configuration you can use again the dialog shown above:

1. Decide which of the series-connected elements should become a representative element of the series configuration.
2. Submit the representative element as if it is the only element between the end nodes of the series combination.
3. When submitting each of the other elements of the series configuration check the In series box and type the name of the element representing the configuration in In series with box.

Statements for submitting an element of series configuration except the representative one should be of the form:

$$name [> type] - series [= expression];$$

where *series* is the name of the element representing the series configuration.



Example. This series configuration of four elements between nodes N13 and N11 can be submitted as

$$E6 \ 13 = 12; \ R6 - E6 = 10; \ L8 - E6 = .2; \ S1 - E6 = (V.13 > 0);$$

11.2.3 Inductive coupling

Inductive coupling between inductors of self-inductances L_a and L_b can be characterized either by means of their **mutual inductance** M_{ab} , or in terms of the **coupling factor**

$$K_{ab} = \pm M_{ab} / \sqrt{L_a L_b}$$

Values of M_{ab} and K_{ab} are positive if the +poles of both inductors represent either the beginnings or ends of the modeled coils, otherwise the values are negative.

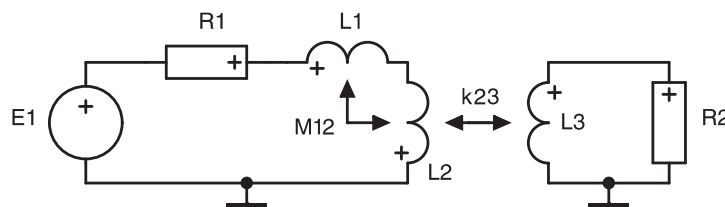
Submitting
text of
mutual
inductance.

In the System menu choose Insert Inductive Coupling

1. In the list type select either M for mutual inductance or K for coupling factor.
2. type the name of the inductive coupling beginning either with M or K in agreement with its type in the Name box.
3. in the Inductor 1 list select or type in the name of the first inductor participating in the coupling
4. in the Inductor 2 list select or type in the name of the second inductor participating in the coupling
5. In the Value box, type the value of coupling parameter in the form of a numeric constant or a symbolic expression.

The resulting statement will be of the form

name [*> type*] *inductor* - *inductor* [= *value*];



Example. There are two inductive couplings in the circuit. the first one is characterized by its mutual inductance and the second one by its coupling factor. The following data demonstrate also the possibility to submit physical elements forming a closed loop as a series configuration. There two such loops in the example. Each of the elements representing a series configuration are connected by both poles to the same node (reference one, in this case).


```

*: Inductively coupled loop circuits
*SYSTEM;
E 0 = 12;      :element representing 1st loop
:remaining elements of the 1st loop
R1-E = .1; L1-E = 10m; L2-E = 15m;
L3 0 = 30m;    :element representing 2nd loop
:remaining element of the 2nd loop
R2-L3 = .15;
:induktive couplings
M12 L1-L2 = -.2; k23 L2-L3 = -.98;
*TR; TR 0 1; PRINT I.E, I.L3, V.L3; RUN; *END;

```

11.3 Submitting texts of basic blocks

The variety of basic blocks is described in Chapter 9. You can insert the block statements in the problem text file using the Insert Basic Block dialog that you can open from the System menu.

:

The 'Insert Basic Block' dialog box contains the following fields and controls:

- Type:** A dropdown menu showing 'explicit block'.
- Name:** A text box containing 'B1'.
- Output node:** A text box containing 'u'.
- Parameter:** A text box containing 'A*sin(omega*time)'. Below this is a row of constants: 10^{15} , 10^{12} , 10^9 , 10^6 , 10^3 , 10^0 , 10^6 , 10^9 , 10^{12} , and π .
- Buttons:** 'Insert', 'Cancel', 'Help', and 'Expression...'.
- Statement:** A text box at the bottom containing the statement 'B1>BS u = A*sin(omega*time);'.

Submitting
text of
basic
block.

1. Choose the type of the submitted block from the type list in agreement with Chapter 9.
2. Type the name of the block in the Name box.
3. Specify the name of the node to which the block output should be connected in the Output node box.
4. Type the numeric constant or symbolic expression characterizing the block in the Parameter box.
5. Click the Insert button and, if asked, enter the required data.

The statement for submitting basic blocks with the exception of the transfer block:

$$\textit{name} [>\textit{type}] \textit{node} [= \textit{parameter}];$$

name is the user-defined name of the block

type is a two-character string denoting the block type in agreement with Table ???. If the first two characters of the name coincide with the block type, the string *>type* can be omitted.

node is a user-defined name of the node to which the block output is connected

parameter is a numeric constant or a symbolic expression defining the block output variable. If the parameter value is 1, the string =1 can be omitted.

Example. The block submitted in the dialog shown above is the explicit block B1 acting as an autonomous source of signal $u = A \sin \omega t$. It will be specified by the statement

$$\text{B1} > \text{BS } u = A * \sin(\omega * \text{time})$$

The *parameter* of a transfer block with the transfer function $F(s) = K \cdot M(s)/N(s)$ is of the following form:

$$[\textit{factor}][*][\textit{numerator}] [/ \textit{denominator}] * \textit{input};$$

factor is a numeric constant or symbolic expression giving the value of the factor K of $F(s)$. The symbolic expression should be enclosed in parentheses (). If the value of K is 1, it can be omitted.

numerator is the name of the polynomial $M(s)$, i.e. the numerator of $F(s)$. If its value is 1, the string *1 can be omitted.

denominator is the name of the polynomial $N(s)$, i.e. the denominator of $F(s)$. If its value is 1, the string /1 can be omitted.

input is the user-defined name of the node, to which the block output is connected

The specifications of the polynomials $M(s)$ and $N(s)$ must precede the specification of the transfer block.

Insert Basic Block

Type: Insert

Name: Cancel

Output node: Help

Input node:

Factor:

10^{-15} 10^{-12} 10^{-9} 10^{-6} 10^{-3} 10^3 10^6 10^9 10^{12} π

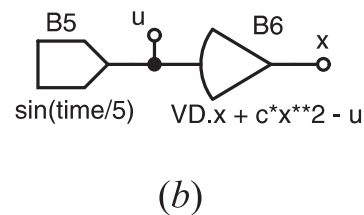
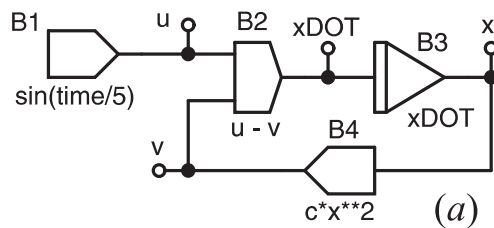
Numerator:

Denominator:

Statement

F>BT Y = (K)*M/N * X;

Submitting
text of
transfer
block.



Examples. Both block diagrams represent the differential equation

$$\frac{d}{dt}x + cx^2 = \sin t/5$$

Text of the diagram in Fig. a

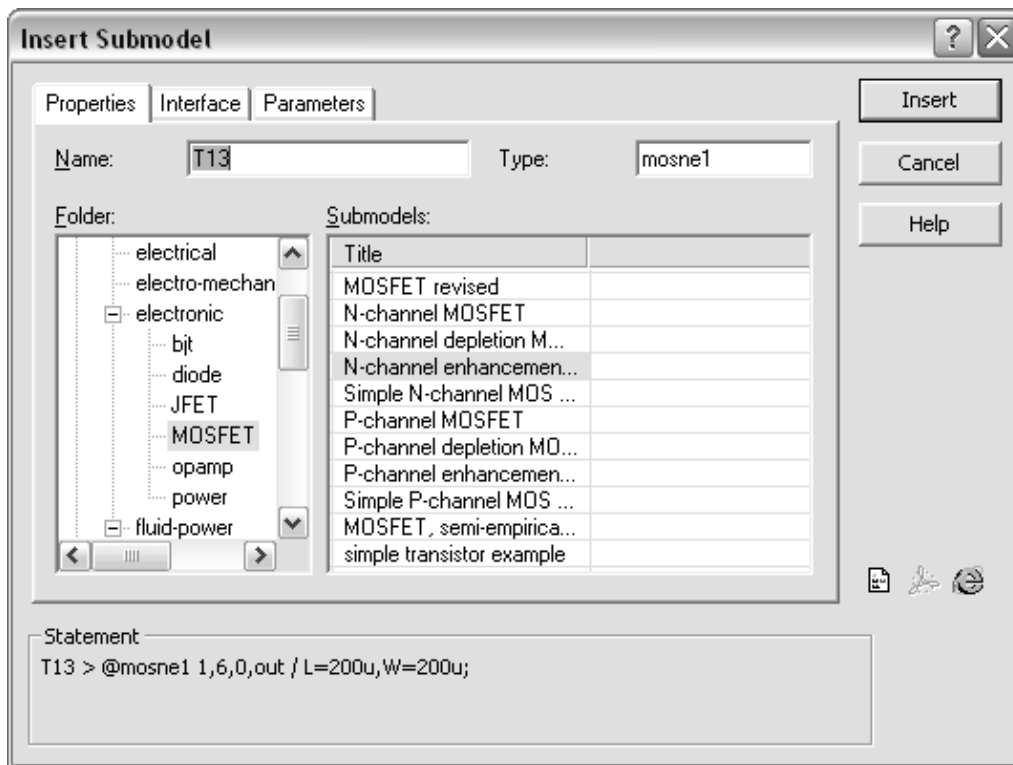
```
*SYSTEM; c = 1m;
B1 > BS u = sin(time/5);
B2 > BS xDOT = u - v;
B3 > BI x = xDOT;
B4 > BS v = c*x**2;
*TR; TR 0 200; PRINT x, u; RUN; *END;
```

Text of the diagram in Fig. b

```
*SYSTEM; c = 1m;
B5 > BS u = sin(time/5);
B6 > BO x = VD.x + c*x**2 - u;
*TR; TR 0 200; PRINT x, u; RUN; *END;
```

11.4 Submitting texts for submodel insertion

The respective dialog opens after choosing Insert Submodel from the System menu.



Submitting
text for
inserting
submodels.

1. When dialog opens
 - either locate in Folder the folder in which the submodel is stored, and then click the submodel title in Submodels,
 - or specify the type of the submitted submodel in the type box.
2. Type the name of the modeled component in the box Name.
3. Open the tab Interface and associate the individual submodel poles with the diagram nodes.
4. Open the tab Parameters and update values of the submodel parameters.
5. Click the Insert button.

The statement for inserting a submodel is of the form

`[name >] @type node [- node...] [/ [parameter =] value [, [parameter =] value...]];`

name is the user name of the component modeled by the submodel

type is the submodel type contiguous to the character @

node is the name of the node, to which the pole is connected (even if this is the reference node identifier 0). Names of the nodes are separated by dashes - or commas , .

parameter is the name of an external parameter of the submodel. The list of the external parameters is separated from the list of nodes by the character /.

value is the numerical constant or symbolic expression giving the value of the external parameter. If the value of an external parameter is missing, the parameter will take over its implicit value specified in the submodel file MOD, or zero value if the default value is not specified there.

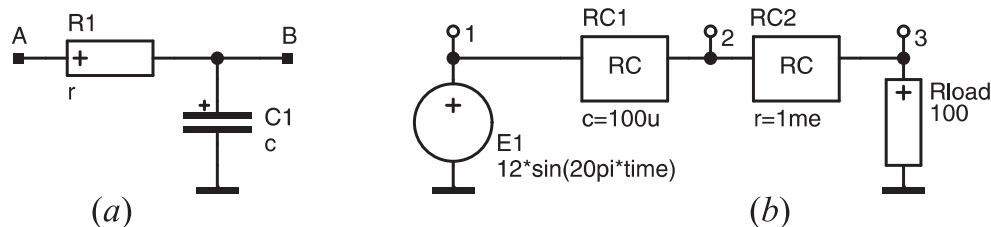
Note: If the external parameters and their values are entered in the same number and order in which they are listed in the MOD file, the strings *parameter* = can be omitted.

Statements in the PRB file to which a submodel has been linked can refer to the following internal submodel variables (these variables cannot be altered from outside of the submodel, however):

- *V.name.node* ... across node variable
- *V.name.element* ... across variable of element G, C or J
- *I.name.element* ... through variable of element R, L, E or S

In all these cases *name* is the name of the submodel instance in the diagram.

Example. Fig. *a* shows the diagram of the submodel RC, Fig. *b* gives an example of exploiting this submodel for modeling components RC1 and RC2 in an electrical circuit. DYNAST printed below the submodel symbols values of external parameters *r* and *c* of the submodel RC specified differently from their implicit values.



Listings of the submodel file RC.MOD and of the circuit file RCRC.PRB:

```
::low-pass filter
RC      ::RC circuit
A,      ::electrical pole
B /     ::electrical pole
r = 1k,  ::[ohm] resistance
c = 1u;  ::[F] capacitance
R1 A-B = r;
C1 B = c;
EO@;

*SYSTEM;
E1 1 = 12*sin(20pi*time); Rload 3 = 100;
RC1 > @RC 1,2 / c=100u;
RC2 > @RC 2,3 / r=1me;
*TR; TR 0 1; PRINT V.3, I.RC1.R1, I.RC2.R1; RUN; *END;
```

Note the references to the internal variables *I.RC1.R1* and *I.RC2.R1* in the command PRINT .

11.5 Catalogue of component parameters

External parameters of submodels can be read in from catalogues of real components in the form of text files with the extension CAT. These files should be arranged in the following way:

```

part
parameter = value[, parameter = value, ...];
part
parameter = value[, parameter = value, ...];
...

```

part is the name (commercial) of a real component

parameter is the name of a submodel parameter

value is a numeric constant or an expression giving a numeric constant

Note: There is no fixed relation between submodels and catalogues. A catalogue is a set of parameter values applicable to an arbitrary submodel. If two different submodels share similar parameters, a common catalogue can be used for both of them. If a catalogue contains the value for a parameter not relevant for the submodel, the value is ignored.

Reference to a catalogue

In the Parameters tab of the Insert Submodel dialog pro submitting submodels in a textual form enter to the Parameters from a catalog box the following text:

```
catalogue.part
```

catalogue is the name of a CAT file with the catalogue data; the file name should be entered without the its path and the CAT extension

part is the name of an component in the catalogue

In this case the statement pro inserting a link to the submodel is of the form

```

component i @submodel interface /
&katalog.part[, parameter = value, parameter = value ...];

```

The submodel link is inserted in a regular way except that the list of parameters begins with the statement &katalog.part. Some of the parameter values specified in the catalogue can be replaced by values specified behind this statement.

Example. The statement

```
T1 i @npl b,e,c / &PARAMS.N1n20,Re=10,Rc=2;
```

refers to data from the catalogue of the component N1n20 stored in the file PARAMS.CAT. Values of parameters R_e and R_c will be replaced, however, by values $R_e = 10$ and $R_c = 2$.

Chapter 12

Nonlinear analysis

Chapter sections

12.1 Modes of nonlinear analysis	12-1
12.2 Submitting nonlinear analysis	12-2
12.3 Fourier analysis	12-10
12.4 Text of nonlinear analysis	12-11

Chapter overview. *Nonlinear analysis allows you to compute transient and steady-state responses of dynamic systems. It also provides location of a quiescent operating point of a nonlinear system, the system linearization and both small- and large-signal analyses at this point. In addition, you can acquire frequency spectra of periodic steady-state responses of nonlinear systems. The nonlinear analysis can be applied to algebro-differential equations in the textual form as well as to physical or block diagrams in the graphical form.*

12.1 Modes of nonlinear analysis

Transient analysis is executed in DYNAST by solving the system of algebro-differential equations

$$\mathbf{f}(\mathbf{x}(t), \dot{\mathbf{x}}(t), t) = \mathbf{0} \quad (12.1)$$

in the interval $t_0 \leq t \leq t_f$ for initial conditions $\mathbf{x}(t_0)$. In general, $\mathbf{f}(\cdot)$ is a nonlinear vector function, $\mathbf{x}(t)$ is the solution vector representing the system transient responses, and $\dot{\mathbf{x}}(t)$ is the vector $\mathbf{x}(t)$ differentiated with respect to the independent variable t .

The objective of the **steady-state analysis** is to compute the values of transient responses $\mathbf{x}(\infty)$, i.e. $\mathbf{x}(t)$ for $t \rightarrow \infty$ after all transient components die out. DYNAST can determine the steady-state responses if they are constant. If, however, they are periodic, DYNAST can provide **Fourier analysis** for computing their frequency spectrum.

To execute the steady-state analysis of the submitted algebro-differential equations (12.1), DYNAST sets automatically $\dot{\mathbf{x}} = \mathbf{0}$ to convert (12.1) into algebraic equations. If the diagram submitted to this analysis represents the dynamic model of a system, DYNAST automatically provides its **conversion into the static model**.

To do this, DYNAST

- ignores all C-type physical elements
- replaces all L-type physical elements by ideal connections
- sets the Laplace-transform variable s in the transfer-blocks BT to zero
- sets the output variables of all differentiating blocks BD to zero
- ignores all integrators BI

Static analysis, i.e. the analysis of static systems, is applied either to nonlinear algebraic equations, or to a static system model. Therefore, both the static and steady-state analysis is executed by solving the set of equations (12.1) in which $\dot{\mathbf{x}} = \mathbf{0}$. It is assumed that the solution vectors \mathbf{x} resulting from these analyses are constant.

In addition, DYNAST can provide the **parameter-sweeping analysis** of the steady-state or static solution. This allows you to compute static characteristics of systems with respect to their parameter swept through an interval. You can even compute families of such characteristics. If the variable t is chosen as the swept parameter, this procedure provides the **quasi-static analysis** dynamic systems.

The constant steady-state solution $\mathbf{x}(\infty)$ represents the **quiescent operating point** of the analyzed dynamic system. DYNAST can use this solution as the vector of initial conditions $\mathbf{x}(t_0) = \mathbf{x}(\infty)$ for a subsequent **large signal transient analysis** in the vicinity of the system quiescent operating point.

The DYNAST procedure for solving the nonlinear algebro-differential equations (12.1) is based on the equation iterative linearization at discrete values of t in the solution interval $t_0 \leq t \leq t_f$. The **last linearized equations** formed during the solution process are in the form

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_{\mathbf{x}=\mathbf{x}_f} \cdot \Delta \mathbf{x}(t) + \left(\frac{\partial \mathbf{f}}{\partial \dot{\mathbf{x}}} \right)_{\dot{\mathbf{x}}=\dot{\mathbf{x}}_f} \cdot \Delta \dot{\mathbf{x}}(t) + \mathbf{f}(\mathbf{x}_f, \dot{\mathbf{x}}_f, t_f) = \mathbf{0} \quad (12.2)$$

where $\mathbf{x}_f = \mathbf{x}(t_f)$ is the **last-solution vector** and $\dot{\mathbf{x}}_f = \dot{\mathbf{x}}(t_f)$.

During its execution, DYNAST temporarily stores the vector \mathbf{x}_f for its use in subsequent nonlinear analysis, where it can be used as the vector of initial conditions $\mathbf{x}(t_0) = \mathbf{x}_f$. DYNAST allows also for storing the vector \mathbf{x}_f in a file that can be used for computations during its later executions (see the commands `SAVE` and `LOAD` in the last paragraph of this chapter).

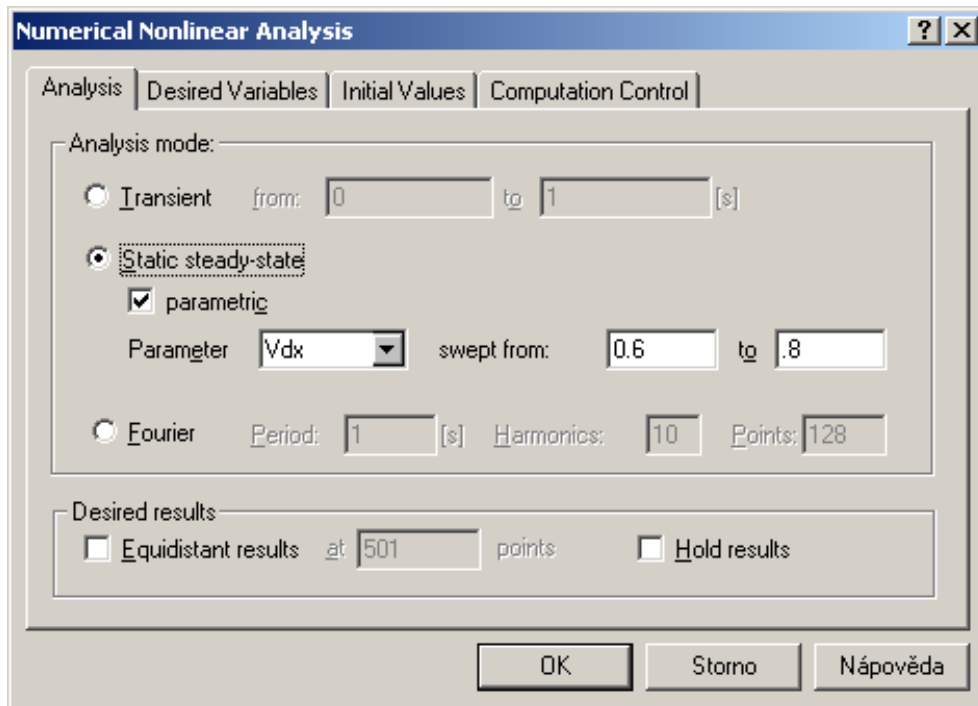
If \mathbf{x}_f corresponds to a steady-state static solution, i.e. if $\mathbf{x}_f = \mathbf{x}(\infty)$, it can be considered as the quiescent operating point of the analyzed system for a subsequent **small-signal analysis** of the nonlinear system in the vicinity of this point. The frequency (Chapter 13) or semisymbolic analysis (Chapter ??) can be then applied for this purpose. The **linearized model** of the analyzed nonlinear system in the vicinity of \mathbf{x}_f is represented by the jacobians shown in (12.2).

12.2 Submitting nonlinear analysis

12.2.1 Transient analysis

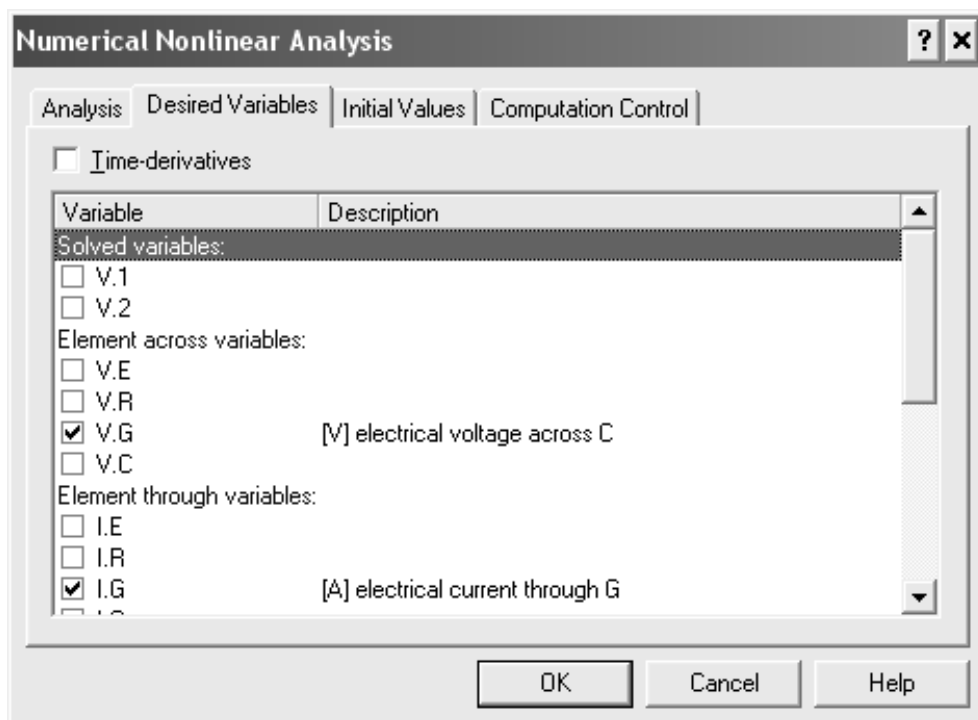
Nonlinear analysis should be applied to the system submitted already in the currently active diagram DIA or problem text file PRB. To specify the transient mode of analysis, from the Analysis

menu, choose Numerical Nonlinear, after which the following dialog opens. As Analysis mode select Transient. In the from and to text boxes specify the lower and upper limits t_0 and t_f of the transient-analysis independent variable t , respectively.



Submitting nonlinear analysis.

12.2.2 Desired variables of nonlinear analysis



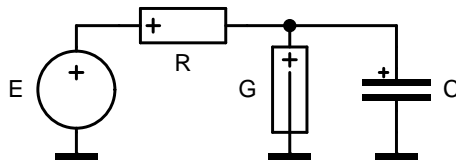
Selecting desired variables for nonlinear analysis.

Click the tab Desired Variables in the dialog opens the list of identifiers of all variables and parameters DYNAST is able to compute for the given systems. The survey of identifiers is shown in Table 3.2. During the analysis computation, DYNAST will store response values

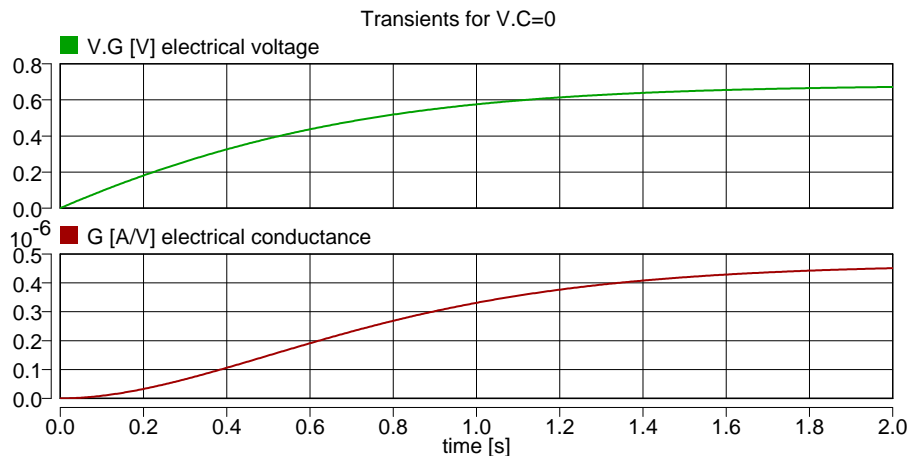
of the checked variables and parameters in the related result O file in a tabular form. The responses can be plotted from this file in different forms (Chapter 15). After checking the box Differentiated variables, you will be able to select also some of the differentiated variables of the system.

After submitting the desired variables click OK and execute the computation. To do this, from the menu Run, choose Run Analysis or Run Analysis & Plot.

Example. Let us consider the electrical circuit with the nonlinear conductor of conductance $G = v_G^2$ S. Parameters of the other circuit elements are: $E = 1$ V, $R = 1$ M Ω and $C = 1$ μ F.



The two above dialogs specify transient analysis of this circuit for $0 \leq t \leq 2$ s with the electrical current i_G , voltage v_G , and conductance G as desired variables. The resulting responses of the last two variables plotted are as follows:



12.2.3 Occurrence of desired-variable points

DYNAST computes the variable values at points unevenly distributed along the independent variable t axis to minimize the computation time while respecting the required accuracy. When the responses are plotted, the computed points are interconnected by line segments. If the underline equations are too 'easy' to solve, the points are sparse and the plotted responses may not be smooth enough. To avoid this, DYNAST can interpolate the unevenly distributed points by points distributed uniformly along the independent variable t axis. These points are then saved in the result file O.

To exploit this option, in the Desired results section of the Nonlinear Analysis dialog check the Equidistant box. In the adjacent text box, specify the total number of the required points in the interval of the independent variable t .

12.2.4 Static or steady-state analysis

To specify the static or steady-state mode of analysis, choose Static or steady-state in the Analysis mode: section of the Nonlinear Analysis dialog.

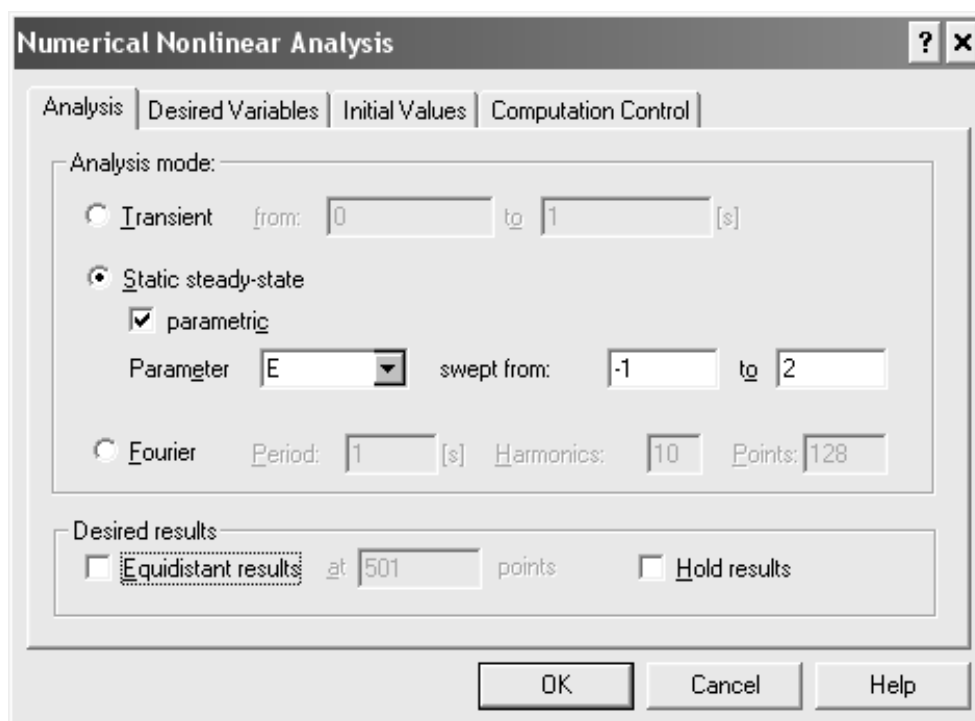
Example. After submitting the static or steady-state analysis for the above circuit, you will be able to find the following lines in the related result file:

```
Steady-state solution
1 ... I.G
2 ... V.G
3 ... G
      1          2          3
3.176703e-007  6.823264e-001  4.655694e-007
```

Thus the resulting variables are: $i_G = 0.3177$ mA, $v_G = 0.6823$ V and 0.4657 mS. Compare the variable values with the values to which the circuit transient responses converge for $t \rightarrow \infty$.

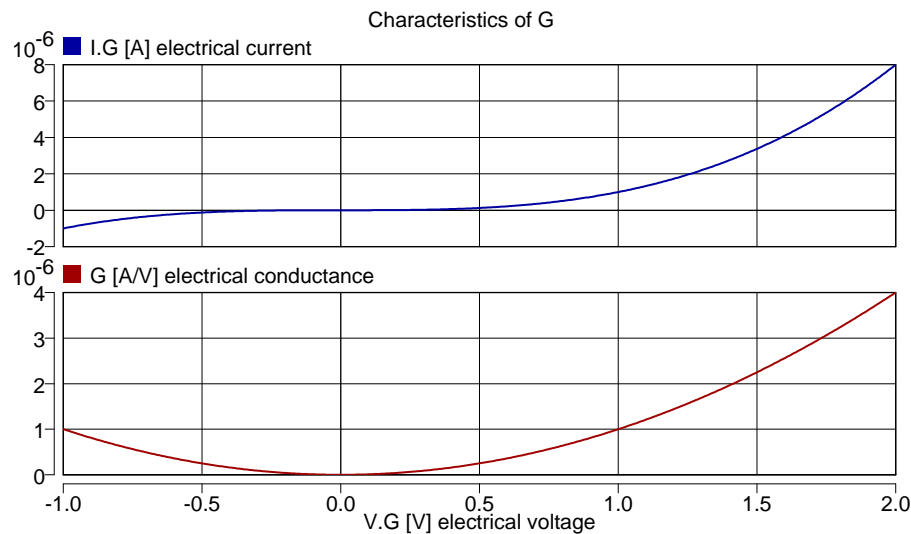
12.2.5 Sweeping-parameter analysis

You can see how the results of the static or steady-state analysis vary with a parameter of the system. To sweep the parameter through an interval, check the Sweeping box in the Nonlinear Analysis dialog. Select then the parameter to be swept and specify the lower and upper limits of its sweeping in the from and to text boxes, respectively.



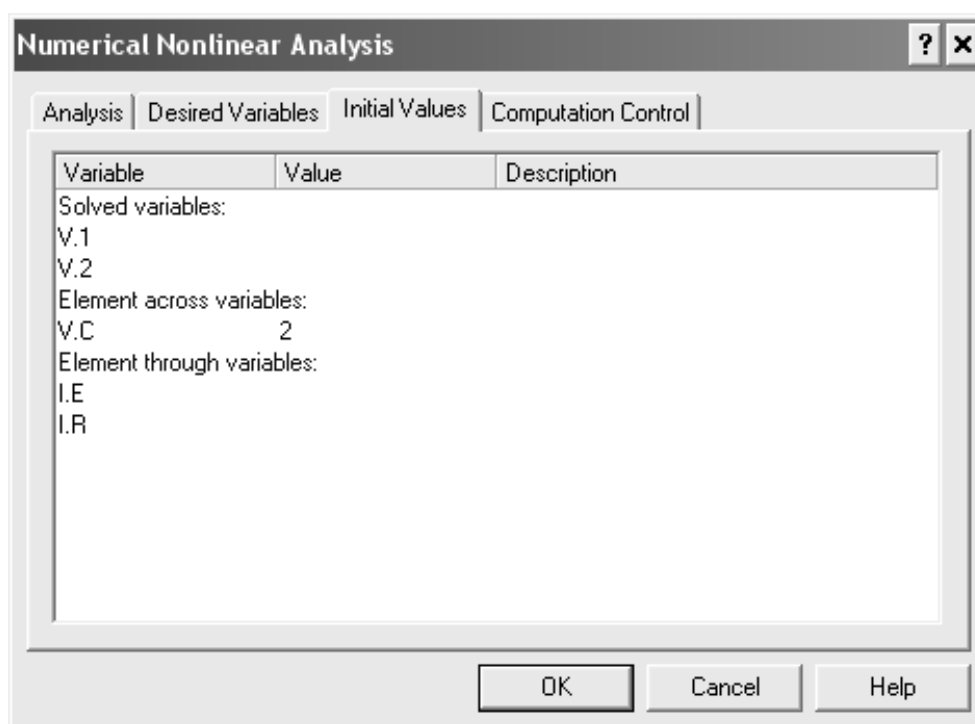
Submitting
parameter-
sweeping
nonlinear
analysis.

Example. Let us set $R = 0$ in our circuit. If we submit sweeping of E through the interval $0 \leq E \leq 2$ V as shown in the dialog, we will receive the following $I_G = f(V_G)$ and $G = f(V_G)$ characteristics of the conductor G :



12.2.6 Initial conditions of transient analysis

The **initial conditions** $\mathbf{x}(t_0)$ in (12.1) are assumed to be zero by default in DYNAST. Nonzero initial conditions can be specified in several different ways. To do this, open the tab Initial Values in the Nonlinear Analysis dialog.



Setting
initial
conditions
for
nonlinear
analysis.

If you want to submit initial-condition values of your own choice, select Clear last solution and enter your values in the tab. Note, that the initial-condition values are specified exclusively for the solved variables in (12.1).

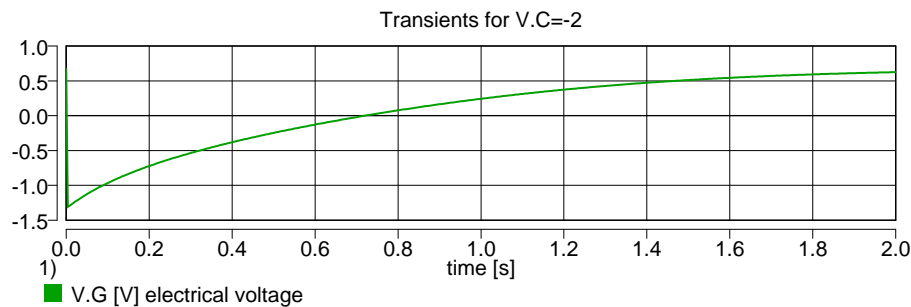
If you are analyzing responses of a system model in the form of a diagram, specify the initial values of

- across variables associated with physical elements of C-type, i.e. capacitors and inertors
- through variables related to physical elements of L-type, i.e. inductors and springs

- output variables of integrator blocks BI

DYNAST will determine initial values of the other variables accordingly.

Example. If, using the dialog, the initial capacitor voltage in the above given electrical circuit is specified as $v_C = -2$ V, the response of v_G will be of the following form:



In some cases you may need the submitted initial-condition values identical with the last-solution values acquired from the previous nonlinear analysis during the same DYNAST run. If this is the case, from the dialog Nonlinear Analysis, open the tab Initial Values and select Leave last solution.

12.2.7 Initial-solution estimate

The Initial Values tab allows you also to specify the initial-solution estimate ${}^0\mathbf{x}$ for the static or steady-state analysis. This option can be utilized to speed up the iteration convergence, or to single out the required solution from the other possible solutions in the case of multiple-solution problems.

Example. The algebraic equation

$$x^3 - 3x + 1 = 0$$

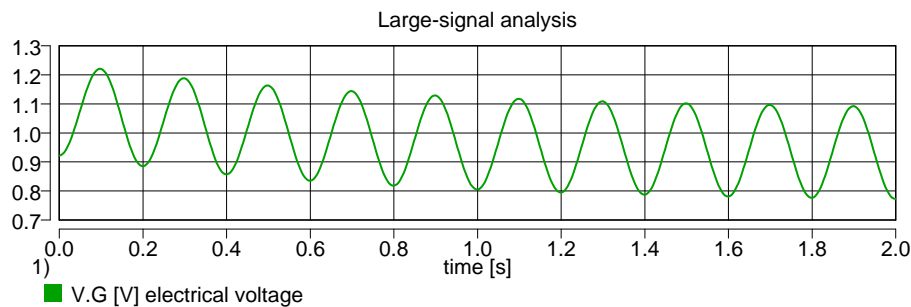
is satisfied for three different values of the solved variable x : ${}^1x = 0.3473$, ${}^2x = -1.879$, and ${}^3x = 1.532$. These values can be determined by repeating the static analysis for three different initial estimates of the solution, for example: ${}^1x^0 = 0$, ${}^2x^0 = -3$ and ${}^3x^0 = 3$.

12.2.8 Large-signal analysis

You might be interested in transient responses to a relatively large excitation in the vicinity of the system quiescent operating point. After submitting the system as well as the excitation, you can specify such a large-signal analysis of the system in two steps:

1. From the Nonlinear Analysis dialog, choose Transient and enter limits of the transient-analysis interval.
2. Open the Initial Values tab and choose Quiescent operating point.

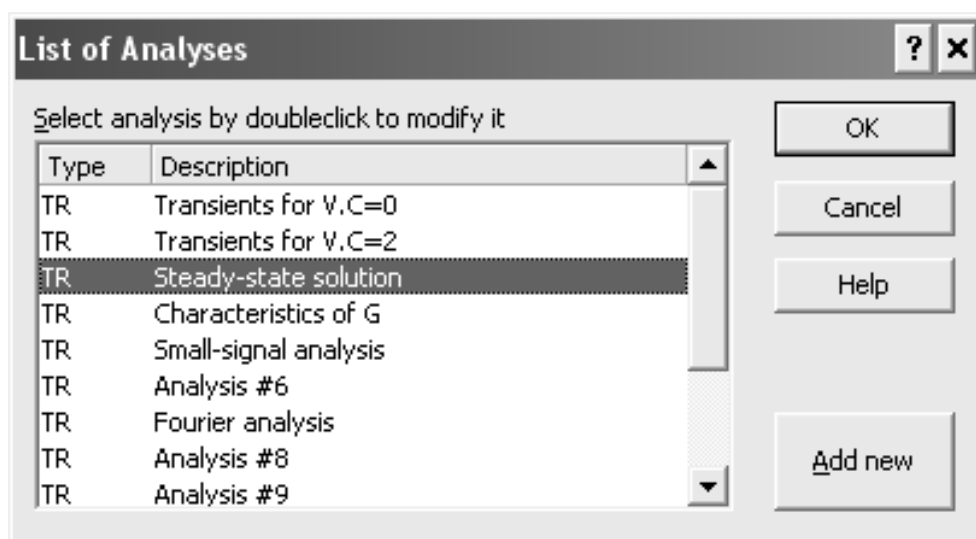
Example. Let us set for this example $E = 1 + 2 \sin 10\pi t$ V in the above given circuit. Then the large-signal response v_G of the circuit in the vicinity of its quiescent operating point is



Note, that the response starts from the operating point identical with the point located earlier by the steady-state analysis.

12.2.9 Changing or adding nonlinear analysis

Let us assume that you have submitted some nonlinear-analysis of a system already and now you want to change it. If you choose Nonlinear analysis from the Analysis menu the following dialog opens.



Changing or adding nonlinear analysis.

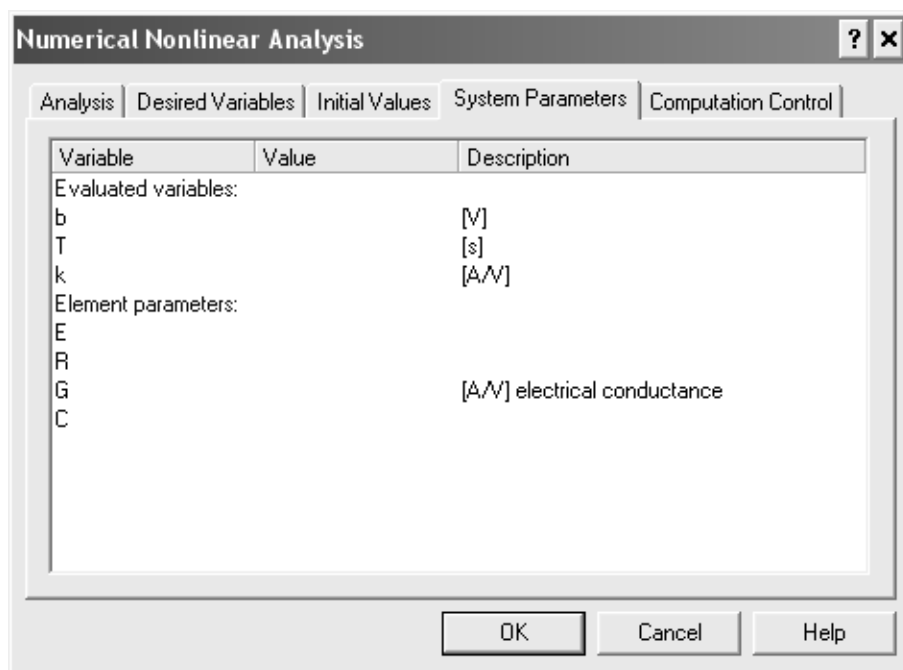
If you want to add another analysis for the same system, click the button New analysis. To change the submitted analysis, select it in the list and click the OK button. As a result, the Nonlinear analysis dialog opens and allows you to make the changes.

12.2.10 Family of responses or characteristics

DYNAST allows you also to store results from several transient analyses or swept static analyses in a common result O file.

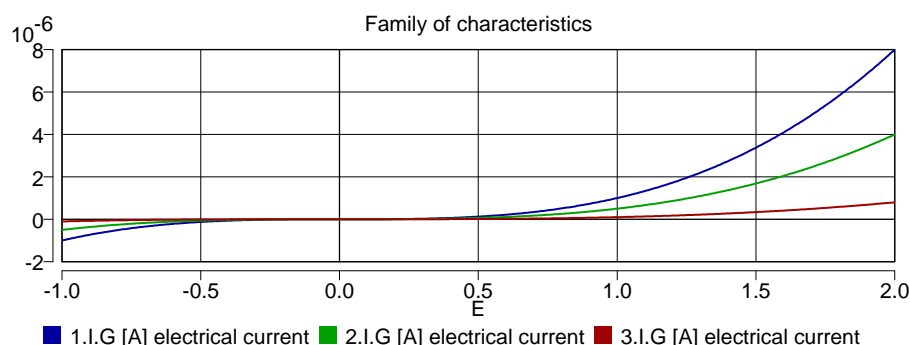
The easiest way in which you can have a family of transient response stored for a set of parameter values in a common result O file is as follows:

1. Specify the desired transient analysis using again the Numerical Nonlinear Analysis dialog. However, before clicking the OK button, check the Hold results box.
2. In each next analysis specification apply the required parameter modification in the System Parameters tab of the Nonlinear analysis dialog. Do not forget to specify initial conditions even if they are zero. At the same time check the HOLD results box.
3. When submitting the last analysis leave the HOLD results box unchecked.

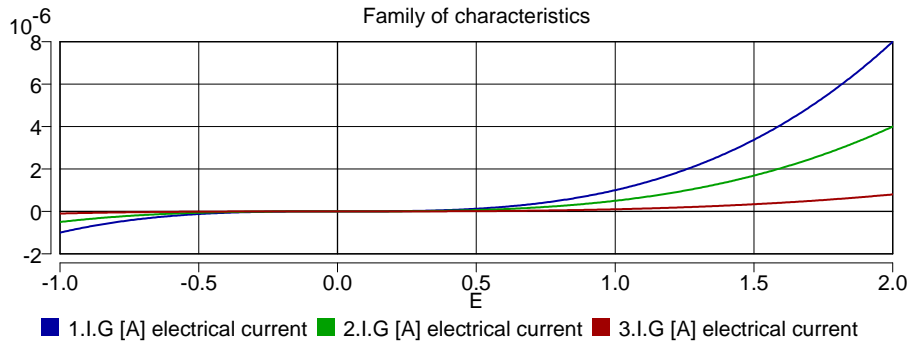


Modification
of system
parameter.

A similar approach can be applied to computation of a family of static characteristics using the swept static analysis this time.



Example. Let us set $R = 0$ in the above given electrical circuit. This allows us to compute the family of three static characteristics $i_g = i_G(v_G)$ of the nonlinear conductor G . The static analysis computed with E swept through the interval $-1 \leq E \leq 2$ V is repeated for three different values of the parameter k : 1, 0.5 and 0.1 μS . The first two analyses should be submitted with the HOLD results box checked.



12.3 Fourier analysis

DYNAST allows you to determine the frequency spectrum of periodic steady-state responses. If $f(t)$ is a steady-state periodic response with the period T , DYNAST can approximate $f(t)$ by the first n members of the Fourier series as

$$f(t) \cong \sum_{k=0}^n A_k \cos(2\pi k/T + \phi_k)$$

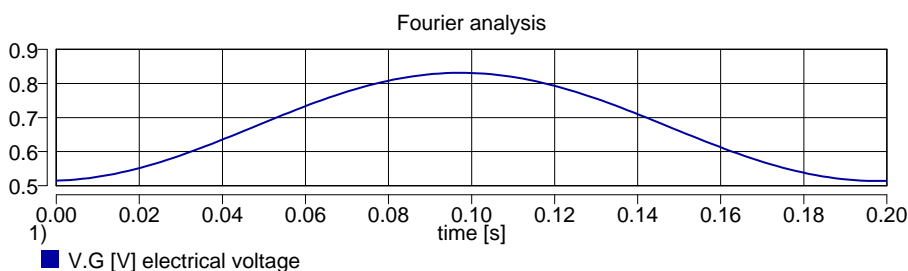
where A_k is the amplitude, and ϕ_k is the phase of the k -th harmonic in the frequency spectrum of the response.

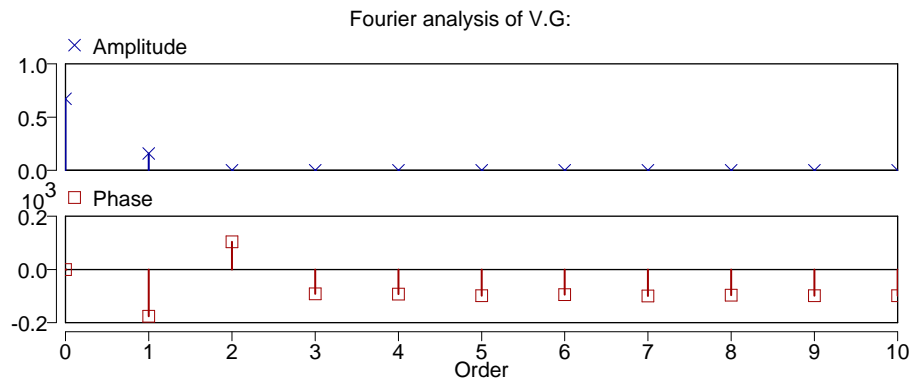
For the Fourier analysis

1. In the section Fourier analysis of the Nonlinear Analysis dialog check Fourier.
2. In the Period text box, specify the period T of the analyzed responses by a numeric constant.
3. In the Harmonics text box, you may set the number n of the harmonics you want to compute ($n = 10$ by default).
4. The text box Samples allows you to change the number of samples (128 by default) DYNAST takes into account in one period of the analyzed response.

DYNAST computes first automatically the analyzed response in the interval $t_f \leq t \leq t_f + T$ starting from the last-solution vector $\dot{x}(t_f)$ from the preceding transient analysis. This waveform can be used for checking whether the periodicity condition $v_G(t_f + T) = v_G(t_f)$ has been satisfied sufficiently.

Example. Let us apply the Fourier analysis to the large-signal response v_G with the $T = 0.2$ s from the previous example. The first plot shows the one-period segment of the response used for the Fourier analysis. The next two plots show the amplitude and phase components of the first ten harmonics of the spectrum. The spectrum lines are emphasized by marks.





12.4 Text of nonlinear analysis

Data for nonlinear analysis are placed in the DYNAST problem text PRB file within the TR section beginning with the *TR; statement. This section must be preceded by the SYSTEM section with the specification of the analyzed problem described by a set of equations (Chapter 6), by a diagram (Chapter 10), or by a combination of both.

The SYSTEM section can be followed by several sections specifying different modes of analysis of the submitted system.

Transient analysis

TR [TIME] t_0 t_f ;

t_0 and t_f are numeric constants or symbolic expressions determining the lower and upper limits of the independent variable t

TIME is the identifier of the variable t that can be omitted

Static or steady-state analysis

DC;

Static analysis with swept parameter

DC [parameter] min max ;

parameter is the identifier of the analyzed system parameter (TIME by default).

min and **max** are numeric constants or symbolic expressions giving the lower and upper limits of the swept parameter

Transient analysis from steady state

DCTR [TIME] t_0 t_f ;

Desired variables

```
PRINT [(points)] [!XALL ,] variables [, variables...];
```

!XALL evokes all solved variables as desired variables

variable is an identifier of the desired variable or parameter. A survey of their identifiers is given in Table 3.2.

points is an integer specifying the number of points specifying the number of the independent-variable points evenly distributed over the analysis interval at which the desired-variable values are interpolated and saved in the result O file. If the string '*(points)*' is omitted, the values are saved at unevenly distributed points at which they have been computed.

Initial conditions of solution estimate

```
INIT variable = value [, variable = value...];
```

variable denotes

- the user-defined identifier of a solved variable
- the user-defined identifier of the across variable of a C-type physical element
- the user-defined identifier of the through variable of an L-type physical element
- identifier of the output variable of an integrator block BI
- **!XALL** allows submitting identical value for all initial conditions
- the command **!XALL** if the initial conditions of all solved variables are to be set to the same value
- the command **!XMAX** to set the value estimate of the solution-vector norm for the computational control
- **!XMAX** , statement for computation control

value is a numeric constant or symbolic expression. During the first analysis it is zero by default. During the next analyses DYNAST uses these values if they have not been erased by the **RESET** command.

Saving the last solution

```
SAVE name[.INIT];
```

name is the name of the file for saving the last-solution vector from the the nonlinear analysis

Loading of initial conditions from file

```
LOAD name[.INIT];
```

name is the name of the file into which a last-solution vector was saved during the previous nonlinear analysis of the current system

Statement canceling and solution erasing

RESET;

Modification of system parameters

MODIFY *parameter* = *value* [, *parameter* = *value*...];

parameter is the identifier of a system parameter

value is a numeric constant or symbolic expression

Fourier analysis

FOUR *period* [, *harmonics* [, *samples*]];

period is a numeric constant giving in seconds the period of the analyzed steady-state periodic responses

harmonics is an integer giving the number of the required harmonics in the resulting spectrum (10 by default)

samples is an integer specifying the number of the response samples considered in its one period during the analysis (128 by default)

If there are transients in the analyzed response, the Fourier analysis must be preceded by transient analysis of a length during which all the transients die out sufficiently.

Analysis execution

RUN [HOLD] [*control* = *value* [, *control* = *value*...]];

HOLD invokes analysis, but its results are not saved into the O file until the first of next analyses is invoked by the RUN statement without the HOLD command

control is the identifier of the parameter computation control

value is a numeric constant giving the control parameter value

Example. Problem text PRB file generated by DYNAST for the electrical circuit analyzed in this chapter.

```
*: Circuit with quadratic conductor
*SYSTEM;
b = 0;    ::[V]
T = .2;   ::[s]
k = 1u;   ::[A/V]
E 1 = 1 + b*sin(2pi/T*time);
R 1-2 = 1me;
G 2 = k*v.G**2;
C 2 = 1u;
```

```

*TR;
TR 0 2;
PRINT(501) I.G, V.G, G;
RUN; ::Transients for V.C(0)=0
TR 0 2;
RESET;
INIT V.C=-2;
PRINT(501) I.G, V.G, G;
RUN; ::Transients for V.C(0)=-2
DC;
RUN; ::Steady-state solution
MODIFY R = 0;
DC E -1 2;
RUN; ::Characteristics of G
RUN HOLD;
MODIFY k=.5u;
RUN HOLD;
MODIFY k=.1u;
RUN; ::Family of characteristics
MODIFY R=1me, b=5, k=100m;
DCTR 0 2;
PRINT (1001) I.G, V.G, G;
RUN; ::Large-signal analysis
FOUR T;
RUN; ::Fourier analysis
*END;
::I.G [A] electrical current
::V.G [V] electrical voltage
::G [A/V] electrical conductance
::V.C [V] electrical voltage

```

Example. The algebraic equation $x^3 - 3x + 1 = 0$ is satisfied for three different values of the solved variable x : $^1x = 0.3473$, $^2x = -1.879$ and $^3x = 1.532$. These values can be found by solving the equation repeatedly for three different initial estimates of the solution, for example: $^1x^0 = 0$, $^2x^0 = -3$ and $^3x^0 = 3$ (see Chapter 12).

```

*SYSTEM; SYSVAR x; 0 = x**3 - 3*x + 1;
*TR; DC; PRINT !XALL; RUN;
INIT x=-3; RUN; INIT x=+3; RUN; *END;

```

Chapter 13

Numerical frequency analysis

Chapter sections

13.1 Excitation for numerical frequency analysis	13-1
13.2 Submitting numerical frequency analysis	13-2
13.3 Text of numerical frequency analysis	13-4

Chapter overview. *The numerical frequency analysis can be applied diagrams with frequency-dependent parameters typical for models with distributed parameters. This is in contrast to the frequency analysis resulting from the DYNAST semisymbolic analysis which is restricted to models with rational transfer functions with constant parameters.*

13.1 Excitation for numerical frequency analysis


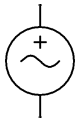
When linear or linearized systems are in a steady state and excited by sources of harmonic (sinusoidal) signals of a certain frequency, their responses are harmonic too. While the frequency of each response is identical with that of the excitation, the response amplitudes and phases are different in general. DYNAST forms for each exciting frequency a new set of complex algebraic equations characterizing the system and solves it numerically.

Harmonic excitation should be applied to the analyzed diagram in the form of one or several special physical elements shown in Table 13.1. This is the source of harmonic through variable (type FJ), and the source of harmonic across variable (type FE). The source variable should be specified in the form

$$p = U @ \varphi$$

where U stands for the excitation amplitude, and φ [rad] for the phase shift. The default values are $p = 1 @ 0$ (the characters @ 0 can be omitted).

Table 13.1: Sources of harmonic excitation.

Type	FJ	FE
Generic	through-variable source 	across-variable source 
Constitutive relation	$I(j\omega) = p$	$E(j\omega) = p$

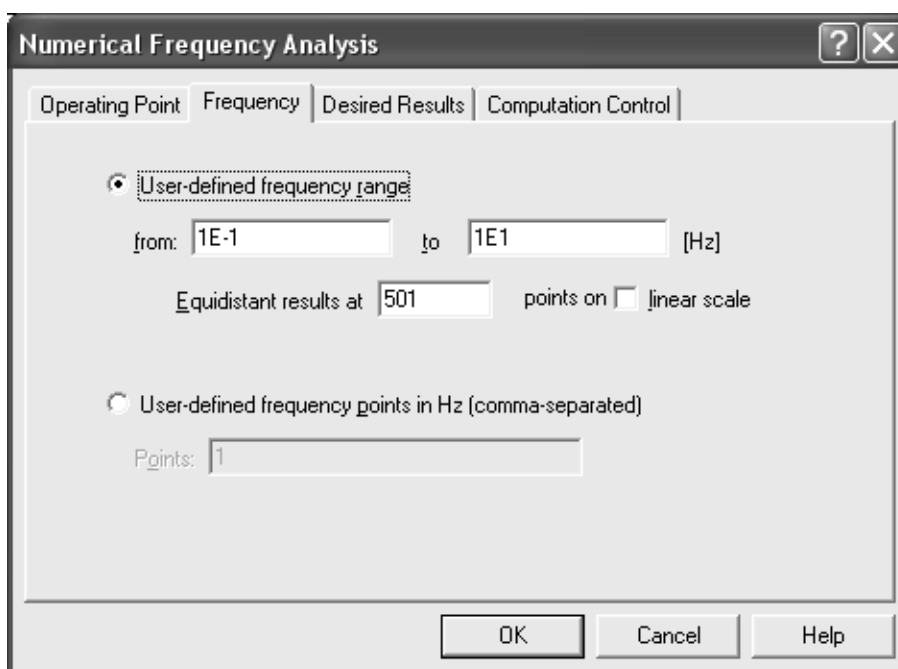
Only harmonic sources are taken into account during the numerical frequency analysis. All the

- through-variable J type sources are ignored
- values of across-variable E type sources are short-circuited
- outputs of explicit BS blocks output variables are set to zero



Example. The figure shows the linear model set up for numerical frequency analysis of a flexibly imbedded machine. The machine is acted on by the harmonic force $F = 15 \sin \omega t$ N. The machine mass is $m = 18$ kg, the bed compliance is $c = 0.3$ mm/N and the bed damping is $d = 200$ N.s/m.

13.2 Submitting numerical frequency analysis



Frequency specification.

After submitting the diagram of your system model, choose Numerical Frequency Analysis from the Analysis menu.

To specify frequencies involved in the analysis, open the Frequency tab from the dialog Numerical Frequency Analysis. Then you will have two options:

- If you choose Frequency range you will be able to specify the lower and upper limits of the frequency range as well as the number of frequency points equidistantly laid out along the logarithmic or linear scale.
- The other option allows you to specify individual frequency points submitted in a growing sequence and separated by commas.

As the frequency-analysis responses are exponential functions, they cannot be displayed in two-dimensional plots unless they are decomposed into some of their components shown in Table 13.2.

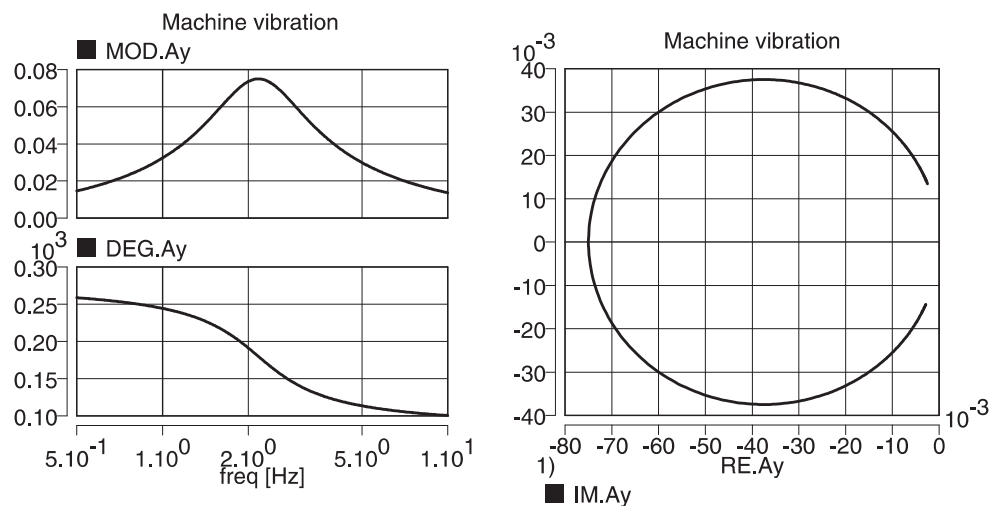
Table 13.2: Components of frequency responses.

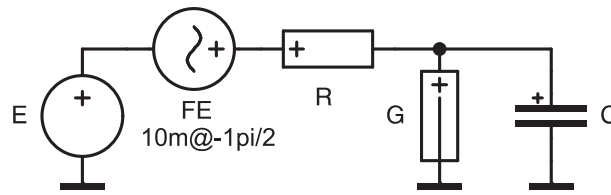
IDENTIFIER	COMPONENT	IDENTIFIER	COMPONENT
MOD	modulus	DB	modulus in dB
RAD	phase in radians	RE	real part
DEG	phase in degrees	IM	imaginary part

Activate either the problem window, or the diagram window in which the system to be analyzed is specified. From the Analysis menu, choose Numerical frequency analysis to open the dialog. After filling in its text fields execute the analysis.

In the next step, open the Desired Variables tab in the Numerical Frequency Analysis dialog. Here you can choose the variables you want to analyze. At the same time specify the variable components you are interest in. Then click OK and execute the analysis.

Example. Frequency characteristics of the vertical velocity of the modeled machine the analysis of which was specified in the dialogs shown above:



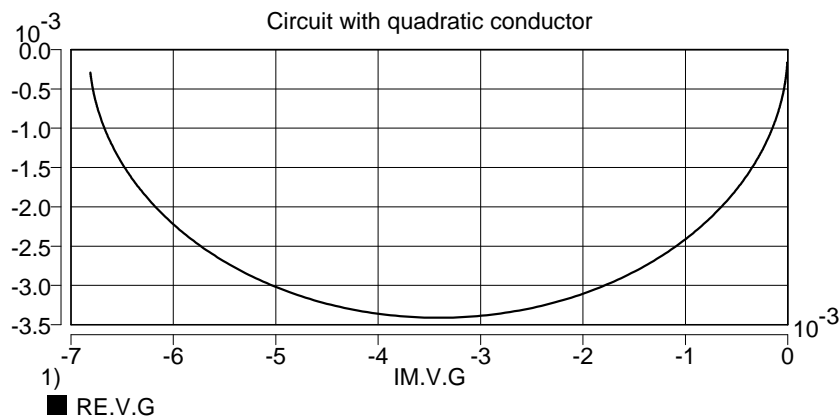


Example. This figure shows the diagram used in Chapter 12 for small-signal transient analysis of a nonlinear electrical circuit. For small-signal numerical frequency analysis a source of harmonic voltage was added to the DC voltage source.

To provide **small-signal frequency analysis** of the nonlinear circuit we must proceed in two steps:

1. Location of the quiescent operating point of the circuit using the nonlinear analysis during which the source of harmonic voltage $u = 10 \cos \omega t = 10 \sin(\omega t - 1\pi/2)$ mV is ignored.
2. Imminent numerical frequency analysis of the circuit linearized during the previous analysis. This time the source of DC voltage $E = 1$ V is ignored.

The resulting frequency characteristic of the voltage v_G can be displayed like this, for example:



13.3 Text of numerical frequency analysis

Specification of the numerical frequency analysis should be included in the AC section of the DYNAST problem text PRB file. The section should start with the `*AC;` statement. This section must be preceded by the `SYSTEM` section with the specification of a diagram that may be combined with equations.

Sources of harmonic excitation

`source [> type] +uzel [- -uzel] [= amplitude [@ phase]];`

or

`source [> FE] - series [= amplitude [@ phase]];`

source is the user-defined identifier of the source

type is the source type, either FE or FJ

+node and **-node** are identifiers of diagram nodes between which the source is placed

series is an identifier of the series element configuration with the source

amplitude is a numeric constant or symbolic expression specifying the source amplitude

phase is a numeric constant or symbolic expression specifying the source phase in radians. If $\varphi = 0$, the string @0 can be omitted.

Frequency analysis

$$\text{FREQ} \text{ } [/[LIN]] \text{ } [min \text{ } max];$$

or

$$\text{FREQ} = f_1, f_2, \dots;$$

min and **max** are numeric constants determining the lower and upper limits of the frequency interval

f_1, f_2, \dots are numeric constants specifying individual frequency points

/LIN, or just **/**, is the string setting the linear frequency scale. If this string is omitted the scale is logarithmic.

If the statement **FREQ** is missing, DYNAST assumes the default frequency range from 10^{-1} to 10.

Required variables

$$\text{PRINT } [(points)] \text{ } component.variable \text{ } [, \text{ } component.variable \dots];$$

points is an integer setting the number of frequencies at which the frequency characteristics should be evaluated (501 by default)

variable is the identifier of the required variable

component is the component identifier in agreement with Table 13.2

Execution of the numerical frequency analysis is activated by the **RUN;** command. Results are stored in an O file.

Example. The statement

$$\text{PRINT } (60) \text{ } MOD.I.R1, \text{ } MOD.A;$$

results in a table giving modules of variables I.R1 and A at 60 points.

Example. Problem text file for the numerical frequency analysis of a flexibly imbedded machine:

```
*:Machine vibration
*SYSTEM;
m > C Ay = 18;      ::[kg] machine mass
d > G Ay = 0.2k;    ::[N.s/m] damping
c > L Ay = 0.3m;    ::[m/N] spring compliance
FJ > FJ Ay = 15@0;  ::[N] excitation force
*AC;
FREQ 5E-1 1E1;
PRINT MOD.Ay, DEG.Ay, RE.Ay, IM.Ay;
RUN;
*END;
::Ay [m/s] machine velocity
```

Example. Small-signal numerical frequency analysis of a nonlinear electrical circuit:

```
: Circuit with quadratic conductor
*SYSTEM;
k = 1u;              ::[A/V]
E 1 = 1;             ::[V]
FE 2-1 = 10m@-1pi/2; ::[V]
R 2-3 = 1me;         ::[ohm]
G 3 = k*v.G**2;      ::[S]
C 3 = 1u;            ::[F]
*TR;
DC;
RUN;
*AC;
FREQ 1E-2 1E1; PRINT RE.V.G, IM.V.G;
RUN;
*END;
```

Chapter 14

Semisymbolic analysis

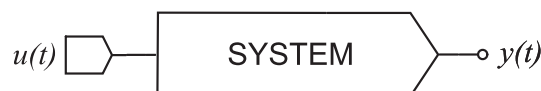
Chapter sections

14.1 Options of semisymbolic analysis	14-1
14.2 Submitting semisymbolic analysis	14-4
14.3 Text of semisymbolic analysis	14-12

Chapter overview. *Semisymbolic analysis can be applied to linearized time-invariant lumped-parameter diagrams. DYNAST provides for the diagrams transfer functions and Laplace transforms of responses to initial conditions in a semisymbolic rational form. Using the inversion of the transforms, DYNAST can then compute semisymbolic-form time-domain responses. These responses and transfer-function frequency characteristics can be also evaluated numerically.*

14.1 Options of semisymbolic analysis

14.1.1 Responses of block diagrams



The figure shows the block diagram of a dynamic system excited by a the source of signal $u(t)$. Assuming that the system is linear with constant parameters, the Laplace transform of its output variable is

$$Y(s) = Y_0(s) + Y_u(s)$$

$Y_0(s)$ is the transform of the **system response to its initial state** $x(t_0)$ while $u(t) = 0$, and $Y_u(s)$ is the transforms of the **system response to the excitation** $u(t)$ when $x(t_0) = 0$. The initial state $x(t_0)$ of a block diagram is related to the initial states of its integrators.

If there are in the block diagram more sources than one, then

$$Y(s) = Y_0(s) + Y_{u1}(s) + Y_{u2}(s) + \dots + Y_{un}(s)$$

where $Y_{ui}(s)$ is the transform of the system response to the i -th excitation $u_i(t)$, when all the other excitations as well as the initial state $x(t_0)$ are zero.

Transforms of responses $Y_{ui}(s)$ to the individual excitations can be expressed as

$$Y_{ui}(s) = H_i(s) \cdot U_i(s)$$

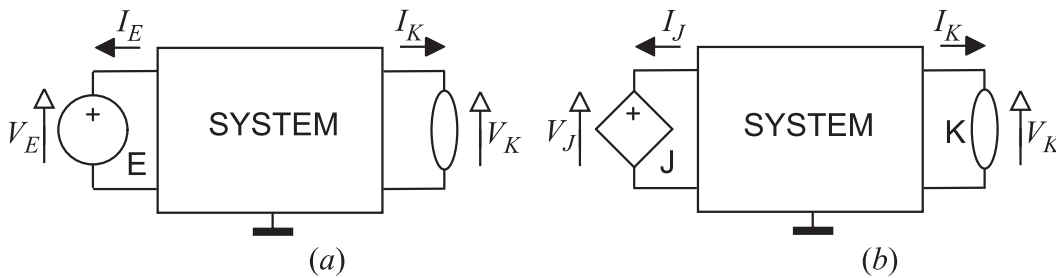
where $H_i(s)$ is the i -th **transfer function** and $U_i(s)$ is the transform of the excitation $u_i(t)$.

14.1.2 Responses of physical diagrams

The formulas for response transforms and their relations to transfer functions mentioned in the previous paragraph are valid for models of dynamic systems in the form of physical diagrams as well. There is a difference in the physical nature of the formula variables, however.

In general, the physical diagrams are excited by independent sources of across or through variables. DYNAST admits also excitation of these diagrams by independent blocks of BS-type which behave as ‘grounded’ sources of across variable.

The response $y(t)$ is represented in physical diagrams by the across or through variable of a physical element. In the examples shown in Fig. *a* and *b* it is the element K, which becomes in the extreme case either the ideal connection (if $V_K = 0$), or the ideal open connection (if $I_K = 0$). The element K can be then replaced by an ideal indicator of through or across variable.



When evaluating $Y_0(s)$, DYNAST considers the following initial values:

- across variables of C-type elements
- through variables of C-type elements
- output variables of BI-type integrators

Table 14.1: Transfer functions of physical diagrams.

Transfer function	Fig. <i>a</i>	Transfer function	Fig. <i>b</i>
admittance	$-\frac{I_E(s)}{V_E(s)}$	impedance	$-\frac{V_J(s)}{I_J(s)}$
transfer admittance	$-\frac{I_K(s)}{V_E(s)}$	transfer impedance	$-\frac{V_K(s)}{I_J(s)}$
across-variable transfer	$\frac{V_K(s)}{V_E(s)}$	through-variable transfer	$-\frac{I_K(s)}{I_J(s)}$

Table 14.1 gives a survey of transfer functions of physical diagrams expressed in terms of across and through variables. The denominator of each function represents there the independent variable of the related excitation source.

When computing the individual components $Y_{ui}(s)$ of the response transform $Y_u(s)$ using the transfer functions $H_i(s)$, DYNAST considers all other

- independent E-type sources as ‘short-circuited’
- independent J-type sources as ‘disconnected’
- independent BS-type blocks as sources of zero signal

14.1.3 Responses in semisymbolic form

In the case of linear models with constant coefficients, the Laplace transform of responses are rational functions of the general form

$$F(s) = K \frac{(s - z_1)(s - z_2) \dots}{(s - p_1)(s - p_2) \dots} = K \frac{a_0 + a_1s + a_2s + \dots}{b_0 + b_1s + b_2s + \dots}$$

DYNAST is able to express the response transforms $F(s)$ in a semisymbolic form with the Laplace-transform variable represented by the symbol s . The multiplicative factor k , the polynomial roots $z_1, z_2, \dots, p_1, p_2, \dots$ (i.e., transform-function zeros and poles, respectively), and the polynomial coefficients $a_0, a_1, a_2, \dots, b_0, b_1, b_2, \dots$ are given there by numbers.

For the semisymbolic-form of rational functions, DYNAST can compute the following responses using the symbolic inverse Laplace transformation:

- **impulse characteristic** of transfer functions $H_i(s)$ as $y_{ui}(t) = \mathcal{L}^{-1}\{H_i(s)\}$
- **step responses** of transfer functions $H_i(s)$ as $y_{ui}(t) = \mathcal{L}^{-1}\{H_i(s)/s\}$
- **responses to initial state** as $y_0(t) = \mathcal{L}^{-1}\{Y_0(s)\}$

Besides these characteristics, DYNAST can also compute time-domain responses $y(t)$ to excitations $e(t)$ the Laplace transform $\mathcal{L}\{u(t)\}$ of which are of the rational form. Some of these functions are given in Table 14.2. The BT-type transfer block can be used as a source of such excitations. Also responses to such excitations and, at the same time, to nonzero initial conditions, are computed by DYNAST.

All these **time-domain responses** can be expressed by DYNAST in the general semisymbolic form

$$y(t) = \sum_i k_i e^{\alpha_i t} \cos(\omega_i t - \phi_i), \quad 0 \leq t \leq t_f$$

where t as well as the exponential and cosine functions are expressed by symbols, whereas k_i, α_i, ω_i and ϕ_i are given by numeric constants.

The **frequency characteristic** of the system transfer function $H(s)$ is expressed simply by substituting $s = j\omega$ into $H(s)$. To display this complex function $H(j\omega)$ in a plane, it must be decomposed into two complementary components like the amplitude and phase characteristics, or the real and imaginary parts of this function.

Besides the semisymbolic form, DYNAST also evaluates the time and frequency responses numerically. It stores the waveforms in the result O files so that they can be plotted in different forms (Chapter 15).

14.2 Submitting semisymbolic analysis

14.2.1 Submitting transforms of responses

To specify response transforms you want to compute for the following diagram in the active window, open the dialog Semisymbolic Linear from the Analysis menu:

The dialog box titled "Semisymbolic Analysis" has three tabs: "Response Transforms" (selected), "Time-Domain Responses", and "Frequency Characteristics".

Under "Response Transforms":

- Transform identifier:** VCO
- Response variable:** I.E
- Transform of response to:**
 - ☐ unit impulse from source (with a dropdown menu showing 'E')
 - ☒ initial state (with an "Open..." button)
- ☒ Add polynomial coefficients

List of transforms

Identifier	Response	Excitation	Coef
G	V.C	E	yes
Yin	I.E	E	yes
VCO	I.E	INIT	yes

Buttons at the bottom: New transform, Remove, OK, Cancel, Help.

Specification
of response
transforms.

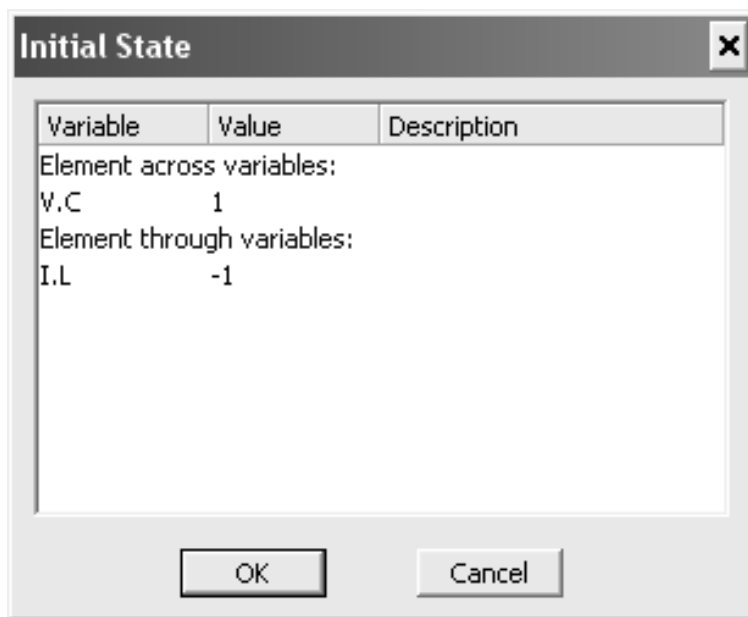
To submit a **transfer function**, follow these steps:

1. In the Function identifier text box, type a name for the function.
2. Choose Input excitation and, in the drop-down list, select a source of excitation for the transfer function definition.
3. In the Output variable drop-down list, select a variable.

4. Check the Polynomial coefficients box if you wish DYNAST to produce coefficients of the function in addition to its roots.
5. Click the New function button in case you want to submit another transfer function, then rewrite the copy of the previous function. Otherwise click the OK button.

To submit the specification of a transform function representing some **response to initial state**, in the second step of the procedure do the following:

Select Initial state, click the Open button and specify the initial values in the Initial State dialog:



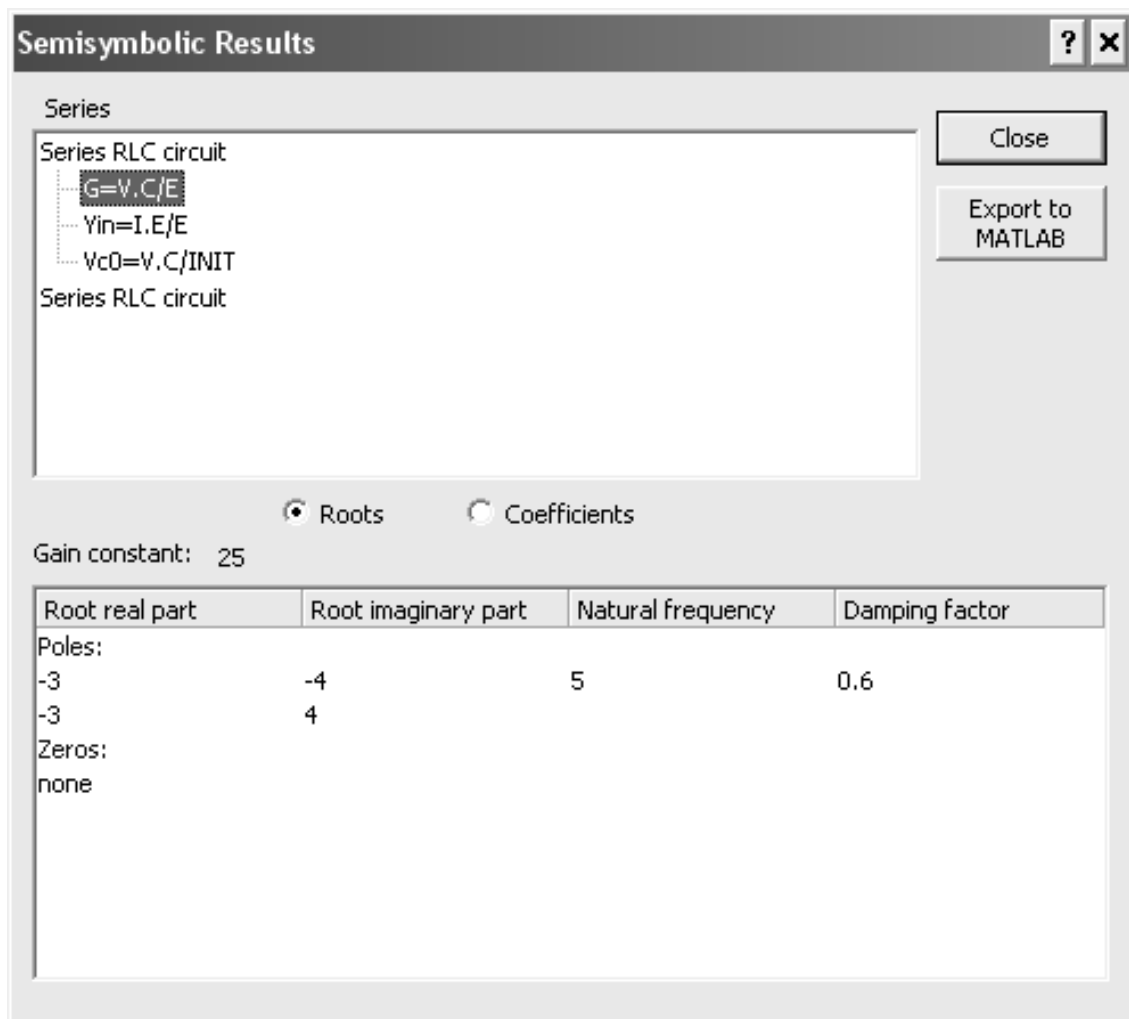
Specification
of initial
state for
semisymbolic
analysis.

After specification of all the required transform functions execute their analysis by choosing Run Analysis from the Run menu.

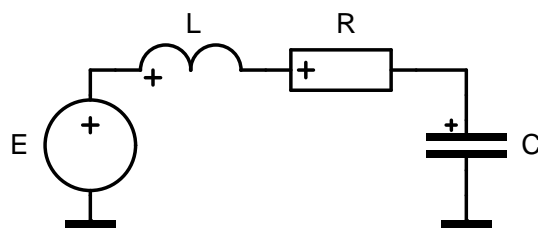
To see the resulting transform functions, do this:

1. From the View menu, choose Result Text.
2. From the View menu again, choose Semisymbolic Analysis Results to open the Semisymbolic Results window.
3. Select an transform function.
4. Choose either Roots or Coefficients to see the related values computed for the selected function.

In the case of a pair of complex conjugate roots $\text{Re } r \pm j \text{ Im } r$, DYNAST computes also the related **natural frequency** $|r|$ and **damping factor** $-\text{Re}/|r|$ besides the real and imaginary parts.



Example. To demonstrate the analysis of transform functions, let us consider the series RLC circuit with these parameters: $E = 20 \sin 50t$ V, $R = 6 \Omega$, $L = 1$ H and $C = 0.04$ F.



The above two dialogs show specification and results for the transform functions

- voltage transfer $G = V_C/E$
- input admittance $Yin = I_E/E$
- transform V_{C0} of the response to the initial conditions $V_C(0) = 1$ V and $I_L(0) = -1$ A.

The resulting transform functions can be rewritten as:

$$G = 25 \frac{1}{(s+3+j4)(s+3-j4)} = 25 \frac{1}{s^2+6s+25}$$

$$Y_{in} = \frac{-s}{(s+3+j4)(s+3-j4)} = \frac{-s}{s^2+6s+25}$$

$$V_{CO} = \frac{s+1}{(s+3+j4)(s+3-j4)} = \frac{s+1}{s^2+6s+25}$$

14.2.2 Submitting time-domain responses

The screenshot shows the 'Semisymbolic Analysis' dialog box with the 'Time-Domain Responses' tab selected. The 'Response variable' is set to 'I.E'. Under 'Response transforms', the 'Transform of response to initial state' is checked, with 'VCO = I.E/INIT' entered in the adjacent field. The 'Response of transfer function' is unchecked. The 'to' section has 'unit impulse' selected. The 'Numerical form of results' is checked. Below, the 'List of time-domain responses' table shows three entries: 'G', 'STEP.G', and 'VCO', all with 'yes' in both 'Semisymbolic form' and 'Numerical form' columns. At the bottom, 'Numerical evaluation' is set to 'Evaluation at: 501' with 'equidistant points within the time interval' selected, and 'determined by DYNAST' is also selected.

Semisymbolic Analysis

Response Transforms | **Time-Domain Responses** | Frequency Characteristics

Response variable: **I.E**

Response transforms

☒ Transform of response to initial state **VCO = I.E/INIT**

☐ Response of transfer function

to ☒ unit impulse ☐ unit step ☒ Numerical form of results

List of time-domain responses

Response transform	Semisymbolic form	Numerical form
G	yes	yes
STEP.G	yes	yes
VCO	yes	yes

New Response **Remove**

Numerical evaluation

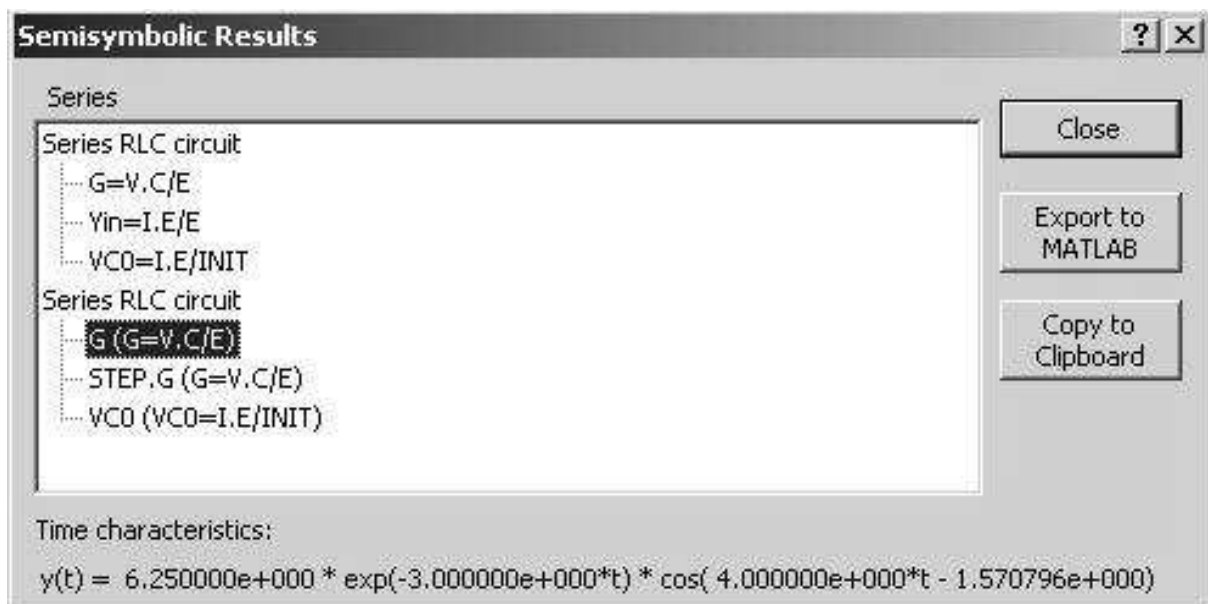
☒ Evaluation at: **501** equidistant points within the time interval

☒ determined by DYNAST

Specification of time-domain responses.

1. Open the tab Time-Domain Responses in the Semisymbolic Analysis menu.
 - To compute **responses to initial state** check Transform of response to initial state and select the required function in the drop-down list.
 - For **transfer-function response** check Response of transfer function and select the required function in the drop-down list. Then choose either unit impulse or unit step for the excitation.
 - To compute a **complete response** submit response to initial state and then response to the unit impulse or step.
 - If you would like to have a response evaluated also numerically for its plotting, check Numerical form of results.
2. If you have required a response also in the numerical form, you can refine your requirement in the Numerical evaluation field:
 - If you do not want to allow DYNAST to determine the time interval of the response automatically (based on the computed roots), you can specify it here as well as the number of points of the response.
 - Alternatively, you may specify the sequence of individual time points of the response.
3. Click OK. In case you want to submit still another response, click New Response.

You will find the resulting semisymbolic-form time-domain responses in the same window as the roots and coefficients of the transform functions. To see a symbolic-form response of a transform function, click the function in the window.

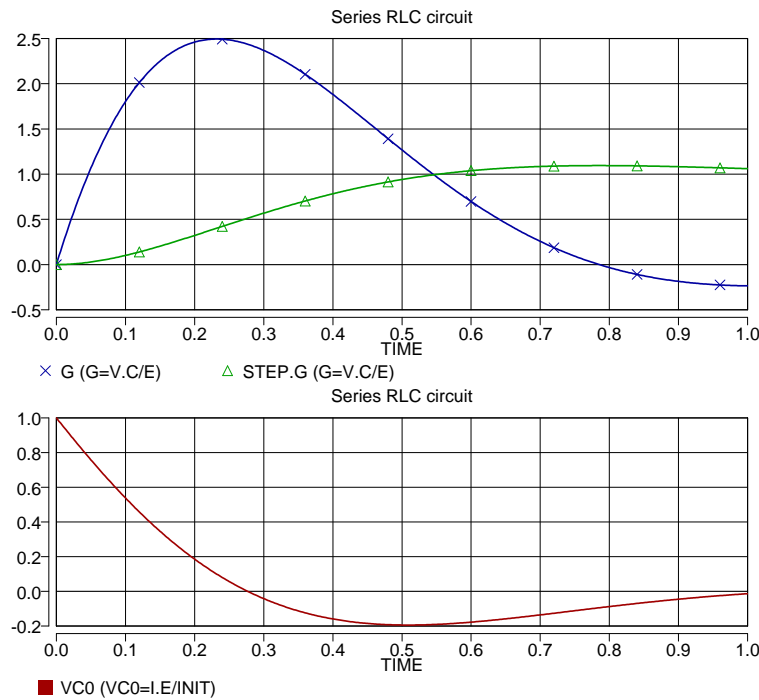


Example. The required time-domain responses of the series RLC circuit as they were computed and plotted by DYNAST:

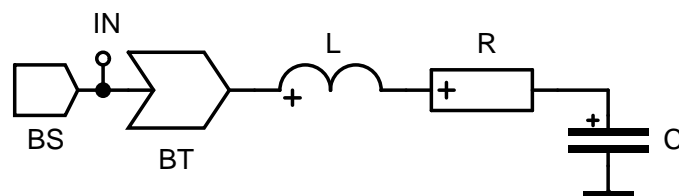
Impulse characteristic of transfer function G : $g(t) = 6.250e^{-3t} \cos(4t - 1.571)$

Step response of transfer function G : $g_{STEP}(t) = 1 + 1.250e^{-3t} \cos(4t + 2.498)$

Response to initial state $v_C(0) = 1$ V, $i_L(0) = -1$ A: $v_{C0}(t) = 1.118e^{-3t} \cos(4t + 0.4636)$



DYNAST allows also for computing system responses to excitations the Laplace transform of which is a rational functions. Some such functions are shown in Table 14.2. A transfer block BT representing the required rational function can be then used as the source of excitation.



Example. Let us consider again the series RLC circuit. Referring to Table 14.2, its excitation by the source of electrical voltage $E = 20\sin 50t$ can be emulated by the BT block with the transfer function

$$\mathcal{L}\{20\sin 50t\} = 20 \frac{50}{s^2 + 50^2}$$

The required time-domain response was computed and plotted with the following results:

$$v_C(t) = 0.2005 \cos(50t + 1.691) + 3.163e^{-3t} \cos(4t + 1.241)$$

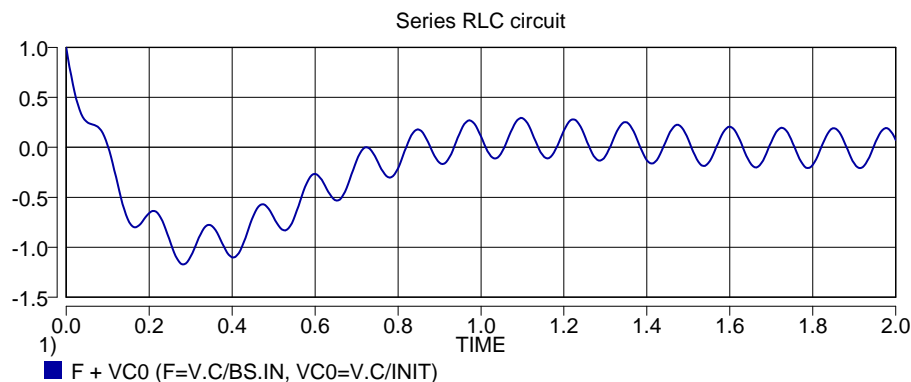


Table 14.2: Obrazy Laplaceovy transformace elementrnch funkc.

Funkce	$f(t) = 0, t < 0$	$\mathcal{L}\{f(t)\}$
Jednotkov impuls	$f(t) = \delta(t)$	1
Jednotkov skok	$f(t) = 1, t > 0$	$\frac{1}{s}$
Rampov funkce	$f(t) = t, t > 0$	$\frac{1}{s^2}$
Exponencieln funkce	$f(t) = e^{-\alpha t}, t > 0$	$\frac{1}{s + \alpha}$
Kosinusovka	$f(t) = \cos \omega t, t > 0$	$\frac{s}{s^2 + \omega^2}$
Sinusovka	$f(t) = \sin \omega t, t > 0$	$\frac{\omega}{s^2 + \omega^2}$
Tlumen kosinusovka	$f(t) = e^{-\alpha t} \cos \omega t, t > 0$	$\frac{s + \alpha}{(s + \alpha)^2 + \omega^2}$
Tlumen sinusovka	$f(t) = e^{-\alpha t} \sin \omega t, t > 0$	$\frac{\omega}{(s + \alpha)^2 + \omega^2}$

14.2.3 Submitting frequency characteristics

1. Open the Frequency Characteristics tab in the Semisymbolic Analysis dialog and select some of the transfer functions in the list.
2. Check one or more frequency-characteristic components for each selected function.
3. Choose frequencies for the analysis:
 - If you do not want to allow DYNAST to determine the frequency interval of the characteristics automatically (based on the computed roots), you can specify it here as well as the number of frequency points of the characteristics.
 - Alternatively, you may specify the sequence of individual frequency.
4. Click OK.

Semisymbolic Analysis [?] [X]

Response Transforms | Time-Domain Responses | **Frequency Characteristics**

Select a transfer function

G
Yin

Components

- ☐ Magnitude
- ☐ Magnitude in dB
- ☐ Phase in radians
- ☐ Phase in degrees
- ☒ Real part
- ☒ Imaginary part
- ☐ Group delay
- ☐ Magnitude slope

Numerical evaluation

☒ Evaluation at: equidistant points within the frequency range

☐ determined by DYNAST

☒ user-defined from: to: [Hz]

Frequency scale

☐ linear ☒ logarithmic

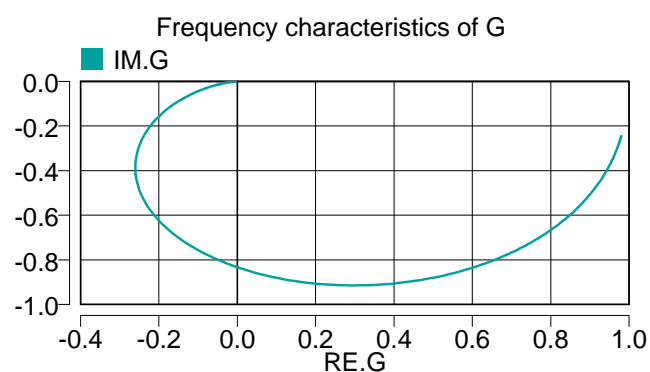
☐ Evaluation at individual points (comma separated)

[Hz]

OK Storno Nápořěda

Specification
of
frequency
characteristics.

Example. The following figure shows the frequency characteristic of the series RLC circuit. The imaginary part of the transfer function G is plotted with respect to the real part of G in the interval from 0.1 to 10 Hz with frequency logarithmic scale.



14.3 Text of semisymbolic analysis

14.3.1 Text of transform functions

Transform functions

should be specified in the PZ section of the PRB file. The section begins with the *PZ; string. The section must be preceded by the SYSTEM section with a diagram specification.

$$\text{TRAN } \textit{function} [, \textit{function} \dots] ;$$

where *function* is of the form: $\textit{identifier} = \textit{response} / \textit{excitation}$ [COEF]

identifier is a user-specified identifier of the required transform function

response can represent

- V. *node*, which is the node across-variable of a node
- V. *element*, which is the across-variable of an element of E, R, L type
- I. *element* the through variable of an element of J, G, C type

excitation, comma separated from *response* by the character '/', may represent:

- the name of a source J or E or of an autonomous block BS assuming that *function* is a transfer function. This can be:
- the string 'INIT' if *function* is the transform of the zero-input response to an initial state

COEF is the string indicating the requirement to compute also polynomial coefficients besides the polynomial roots.

When the string 'INIT' is specified, the initial state is submitted by the separate statement

$$\text{INIT } \textit{variable} = \textit{value} [, \textit{variable} = \textit{value} \dots] ;$$

variable can represent:

- V. *element*, which is the across-variable of a C-type element
- I. *element* which is the through-variable of an L-type element

value is a numerical constant

Transform-function responses

is specified in the TRA section of the PRB file. The section begins with the *TRA; string. The section must be preceded by the PZ section with the specification of transform functions.

$$\text{SYMB } \textit{expression} [, \textit{expression} \dots] ;$$

where *expression* is of the form

$$[\textit{excitation}] \textit{transfer} [+ \textit{state}] \quad \text{or just} \quad \textit{state}$$

excitation specified as `STEP`. indicates excitation by the step function, it is the Dirac function by default

transfer is the identifier of a transfer function from the PZ section

state is the identifier of a transform an initial state response from the PZ section

To evaluate numerically the time responses at specific time points, the points can be submitted either by the statement

`TIME min max;`

or by the statement

`TIME = t1, t2, ...;`

min and **max** are numeric values of the time interval limits

t_1, t_2, \dots are numeric values of the time points

If this statement is missing, DYNAST determines the time interval automatically.

Transfer-function frequency characteristics

is specified in the FREQ section of the PRB file. The section begins with the `*FREQ;` string. The section must be preceded by the PZ section with the specification of transfer functions.

Computation of frequency characteristics of the semisymbolic transfer functions acquired in the PZ section can be specified in the FREQ section.

Frequency is specified either as

`FREQ [/LIN] [min max];` or as `FREQ = f1, f2, ...;`

min and **max** are numerical limits of the frequency interval

f_1, f_2, \dots are numeric values of individual frequency points

`‘/LIN’` is setting a linear frequency scale, it is logarithmic by default

If the statement `FREQ` is missing, DYNAST determines the frequency range automatically.

Table 14.3: Components of frequency characteristics

IDENTIFIER	COMPONENT	IDENTIFIER	COMPONENT
MOD	modulus	DEL	group delay
DB	modulus in dB	SLO	modulus slope
RAD	phase in radians	RE	real part
DEG	phase in degrees	IM	imaginary part

Tabular output for various components of the frequency characteristics is specified as

`PRINT [(points)] component.function [, component.function...];`

The identifiers *component* are listed in Table 14.3 .

Example. Data generated by DYNAST for computing transforms of the series RLC circuit and their time-domain and frequency characteristics:

```

*: Series RLC circuit
*SYSTEM;
omega = 50;
A = 20;
E 1 = A*sin(omega*time);
L 1-2 = 1;
R 2-3 = 6;
C 3 = 0.04;
*PZ;
TRAN G=V.C/E COEF, Yin=I.E/E COEF, VC0=I.E/INIT COEF;
INIT V.C=1, I.L=-1;
RUN;
*TRA;
PRINT(5001) G, STEP.G, VC0;
SYMB G, STEP.G, VC0;
RUN;
*FRE;
FREQ 1E-1 1E1;
RUN;
*END;

```

Example. Data generated by DYNAST for computing time-domain response in symbolic and numerical form of the series RLC circuit to an excitation and initial conditions:

```

*: Series RLC circuit
*SYSTEM;
BS IN = 1;
omega = 50;
denom /POLY/ omega**2,0,1;
numer /POLY/ omega;
BT 3 = 20*numer/denom * IN;
L 3-1 = 1;
R 1-2 = 6;
C 2 = 0.04;
*PZ;
TRAN F=V.C/BS.IN, VC0=V.C/INIT;
INIT V.C=1, I.L=-1;
RUN;
*TRA;
TIME 0 2;
PRINT F + VC0;
SYMB F + VC0;
RUN;
*END;

```


Chapter 15

Plotting results

Chapter sections

15.1 Forms of plotting	15-1
15.2 Scales of plots	15-2
15.3 Arrangement of displayed plots	15-4
15.4 Reading waveform coordinates	15-5
15.5 Import, export and printing of graphs	15-6

Chapter overview. *Your computation results from different analyses saved by DYNAST into an O file in a tabular form can be plotted in various graphical forms. DYNAST allows also for additional rearranging of the resulting plots, for printing and exporting them in different formats as well as for comparing them with data from external sources like digital measuring instruments, for example.*

15.1 Forms of plotting

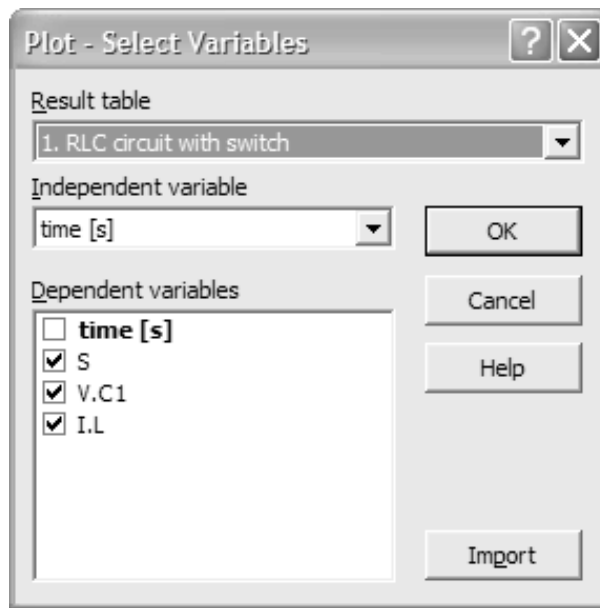
15.1.1 Selection of variables

If you wish to plot analysis results immediately after their computation, from the Run menu choose Run Analysis & Plot, or click the corresponding icon in the toolbar of the main window.

In case you want to plot variable waveforms computed and saved in an O file earlier, open first the window with the O file containing the analysis results in the textual form. Instead, you may open either the diagram or textual window with the specification of the analyzed system. Then choose Result plot from the View menu.

In all cases, a window with the plotted waveform of the first desired variable from the first analysis these results are saved in the related O file will open.

To plot waveforms of other analyzed variables choose Set variables from the Plot menu, or right click the plotted graph. The following dialog opens afterwards.



Selection of variables for plotting.

1. In the Result table list, select one of the tables saved in the resulting O file.
2. In the Independent variable list, select the variable to be considered as the independent variable associated with the horizontal axis of the plot (it can be different from the independent variable of the analysis).
3. In the Dependent variables list, select the variables you want to have displayed as dependent variables along the vertical axis.

15.1.2 Plots in several windows

You may display in several windows plots of results from one or several analyses saved in the common O file. If you have displayed a plot in a window, choose **New Window** in the **Window** menu, or right click the plotted graph. As a result, a new window opens with a copy of the previous plot. Then select the independent and dependent variables for the plot in the new window following the procedure you used for the plot in the first window.

15.2 Scales of plots

15.2.1 Automatic setting of scales

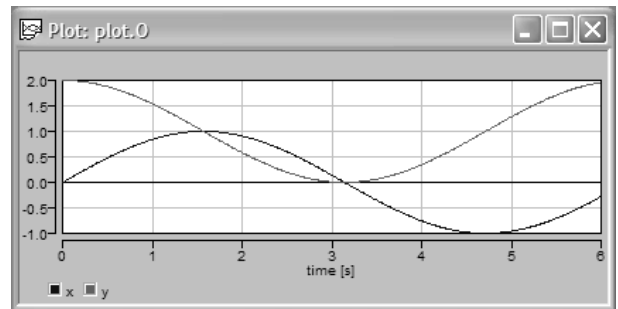
If you select several dependent variables, they will be plotted by default with a common scale along the horizontal axis and, most often, with different scales along the vertical axis. DYNAST adjusts automatically the vertical-axis scales for each variable in a way exploiting the graph area as best as possible. If such an arrangement does not suit to your needs, you can choose other arrangements of the vertical-axis scales to be provided automatically. You may have even each dependent variable plotted in an independent graph, but in the same window.

The different arrangements as well as the corresponding **Axes** menu commands and main-window toolbar icons are shown in the next page.



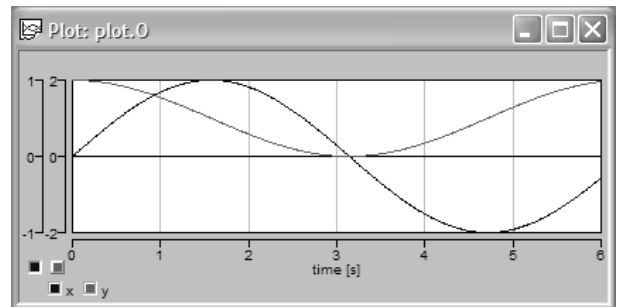
Common Y

Plotting of all selected variables with a common scale on the vertical axis.



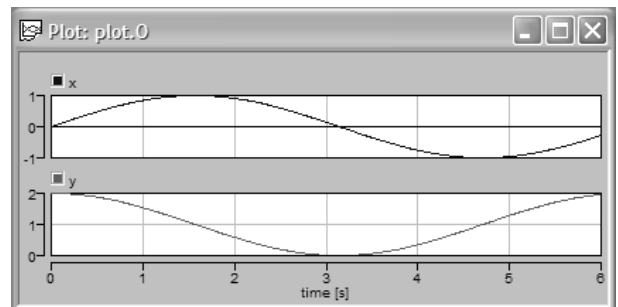
Zero offset Y

Plotting of all selected variables with the common origin of the vertical scale.



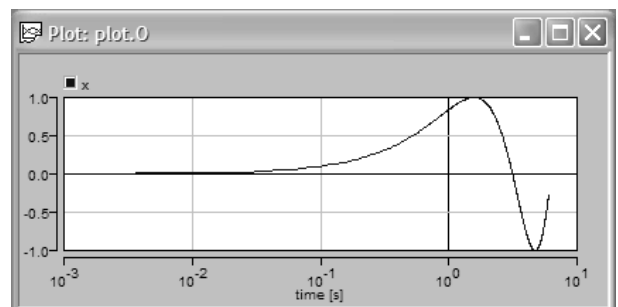
Multiple Y

Plotting each selected variable in an independent graph.



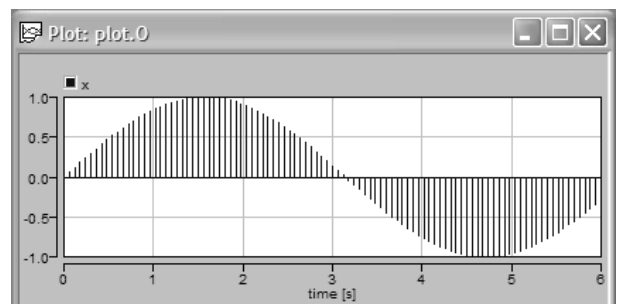
Log X

Plotting of all selected variables with the logarithmic scale on the horizontal axis.



Discrete X

Plotting the selected variables in the form of a histogram.



15.2.2 Setting scales by the user

By default, the graph scales are chosen automatically in a way fully exploiting the area available for the graph. A more detailed plotting of only a part of the graph you can achieve in either of two ways:

- Drag the mouse with its left button pressed across the required part of the graph diagonally. This part selection can be canceled by choosing Undo zoom from the Axis menu. You can return to the detailed plotting of selected graph part by choosing Redo zoom.
- Choose Custom Range from the Axes menu. The dialog for individual setting of the scale ranges for the graph variables. The ranges can be returned to their original state by choosing Full View.

15.3 Arrangement of displayed plots

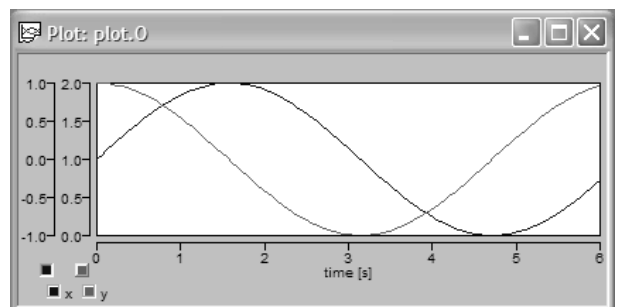
15.3.1 Graph grid and variable waveform marks

There are the following commands in the Plot menu and main-window toolbar icons for the plot arrangement:



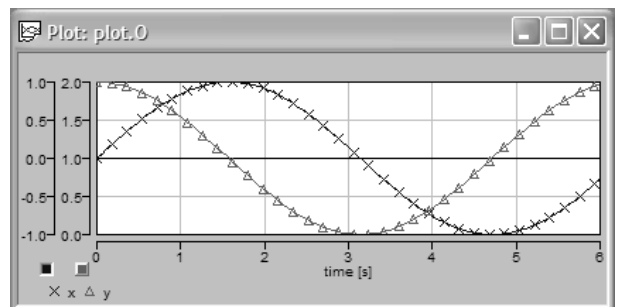
Show Grid

Display or removal of the graph grid.



Point Marks

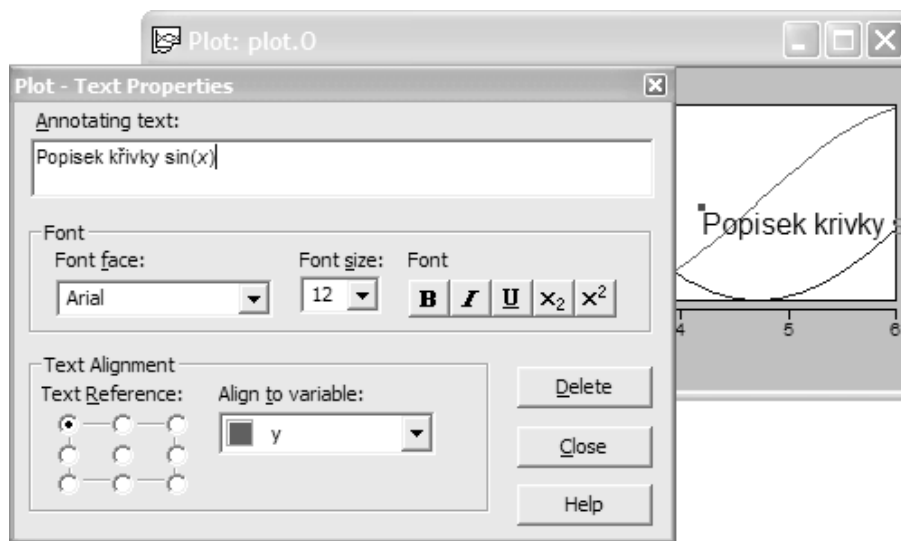
Differentiation of displayed variable waveforms by marks.



The marks are useful especially when the plotted waveforms are printed in the black-and-white form. The mark density can be controlled at the right end of the main-window toolbar.

15.3.2 Notes in graphs

The displayed variable waveforms may be commented by textual notes. To do so, use the dialog Text properties opened by choosing Text from the Plot menu.



Properties
of textual
notes in
graphs

The note text should be typed in the Annotating text field. You can choose for the selected note text specific fonts (bold, italic, underline, subscript, superscript).

If you want the text to follow a variable waveform when its graph is transformed, utilize the Text Alignment field:

1. Select one of the variables in the Align to variable roll-down menu.
2. Drag the reference point displayed together with the text to an appropriate place on the variable waveform.
3. Select one of the radio buttons in the Text Reference field to determine the note position with respect to the reference point.

After inserting all notes click the Close button. The note position can be modified by mouse even after closing the dialog. By clicking a note you can open the dialog again to correct the note or to remove it by the Delete button.

15.4 Reading waveform coordinates

Coordinates of the displayed variable waveforms can be read in the following way:

1. Choose Curve Tracing from the Plot menu to open the Cursor window.
2. Point the waveform the coordinates of which you want to read by moving the cursor to it without clicking. The coordinates will appear in the Cursor window. When you move the mouse along the horizontal axis the cursor will follow the chosen waveform. Precision of the coordinate reeding increases if the Shift key is pressed.
3. To select another waveform, right click the mouse and then click the new waveform.
4. The displayed coordinates are related to the origin defined by the plot scales. If you wish to choose another coordinate origin, choose Reference Cursor from the Plot menu and click the new position of the origin in the graph.

5. If there are more variables with different scales plotted in the graph, in the window rim select the name of the variable the scale of which should be considered, or click the vertical axis of the variable.
6. The cross of axes as well as the reference coordinate system can be removed by a right click. Another right click cancels the regime of coordinate reading completely.

15.5 Import, export and printing of graphs

15.5.1 Common plotting of different graphs

To import a variable waveform from another table in the same or different O file into the current graph follow these steps:

1. Choose Set Variables from the Plot menu and click the Import button.
2. Select the O file, from which the required variable should be imported. Then click the Open button.
3. In the dialog Plot - Select Plot to Import select the imported table in Result tables list.
4. In the Independent variable list, select the variable with respect to which the other variables should be plotted. Then click OK to confirm the selections.
5. Choose Set Variables from the Plot menu to open the dialog Plot - Select the Variables. The imported variables will appear there in the bottom of the Dependent variables list. The independent variable chosen above can be changed here by rightclicking another variable and by choosing Independent.
6. The imported variables can be removed by rightclicking any of them in the list and choosing Remove Plot.

Scales of the variable axes in the resulting graph are adjusted automatically in such a way that the waveforms of all variables are displayed in their full range. It is so even if the waveforms in the two tables differ by their ranges as well as by the number and distribution of their computed points.

15.5.2 Plotting data from external sources

The data plotted simultaneously from different O files do not need to be generated by DYNAST exclusively. They can come from other sources like digital measuring instruments, for example. This option allows for comparing the computed and measured variables.

Correct plotting of data requires their conversion into the DYNAST O file format:

1. Row beginning with the character # followed by the table title
2. Empty row
3. Row with the independent variable description in the form

X ... *variable name*

4. Rows with descriptions of dependent variables, each in the form

i ... variable name

where *i* is the serial number of the variable

5. Empty row

6. List of numerical values of the variables in the form

mantissa[\pm exponent]

where mantissa must include the decimal point

7. Row beginning with the character #

If the total number of variables is n , and the waveform of each variable is given by r numbers, the total number of variable values in the table should be $r \times n$. If it is not so, values forming the last incomplete row are ignored.

Example. An O file with waveforms of four interdependent variables may be arranged in the following way:

```
# Title

X ...    variable_1
1 ...    variable_2
2 ...    variable_3
3 ...    variable_4

0.000000e+000  0.000000e+000  0.000000e+000
               0.000000e+000
6.531626e-001  8.866773e+000  1.133227e-003
               4.354418e+000
1.019687e+000  9.033002e+000 -9.033002e-003
               6.666610e+000

#
```

Plotting data in an O file can be done in two steps:

1. Choose Open from the File menu, select DYNAST Result Text Files (O file) from the Files of type: roll-down menu and specify the path to the target O file to open window with the file in its textual form.
2. Choose Result Plot from the View menu to open window for plotting the O file content.

15.5.3 Printing and exporting plots

For printing a plot by the printer, choose Print or Print Preview from the File menu. By default, plots are printed in the full page size with the portrait orientation. To change this, choose Print Scaling from the File menu.

Plots can be exported for their processing by others programs in the following ways:

Saving to clipboard (for MS Paint or MS Word, for example) – choose Copy from the Edit menu. The plot size corresponds to the size of the plot window.

Export to BMP format (Bitmap) – choose Export to Bitmap from the File menu. The plot size corresponds to the size of the plot window.

Export to EPS format (Encapsulated Postscript) – choose Export to PostScript from the File menu. The plot size, curve thickness and color as well as other features can be set in a dialog.

Saving to clipboard in a textual form (pro further processing in MS Excel, for example) – choose Copy as Text from the Edit menu.

15.5.4 Saving plot layouts

The chosen arrangement of plots and other windows can be saved before their closing in an LAY file. To do so, choose Save Screen Layout from the File menu. Besides the overall arrangement of open windows in the main window, chosen variables including their scales and text notes will be saved in the file. The window alignment during their arrangement is easier if the Ctrl and Shift keys are pressed simultaneously.

A saved window layout including all the plot arrangements can be restored any time on the screen by choosing Load Screen Layout in the File menu.

Chapter 16

Creating submodels

Chapter sections

16.1 Submodel diagrams and equations	16-1
16.2 Submodel text files	16-3
16.3 Libraries of submodel symbols	16-7
16.4 Organization of submodel files	16-11

Chapter overview. *Working environment DYNAST Shell allows you complementing the DYNAST variety of submodels by your own dynamic models by various components of real systems or physical phenomena including their graphical symbols. Dynamic behavior of submodels can be characterized by diagrams consisting of physical elements, blocks, submodels or equations.*


16.1 Submodel diagrams and equations

16.1.1 Submodel diagrams

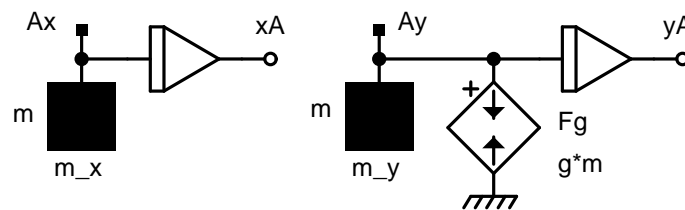
If you want to characterize the submodel dynamics by a diagram follow these steps:

1. Choose New from the File menu and select Diagram in the File type roll-down menu.
2. Enter submodel File name corresponding to the submodel type. At the same time, check the dialog box Submodel.
3. Indicate the modeled real component or phenomenon in the Title field.
4. In the Template section, select Standard template unless you want to document the created submodel.
5. Click OK.

When drawing the submodel diagram in the opened window, you should proceed in the same way as in the case of a system diagrams with the following exceptions:

- Denote by special labels the submodel nodes representing interactions of the modeled component with the rest of the system via the submodel poles. To do so, either choose Pole label from the Place menu, or click the toolbar icon .
- Click each pole label and enter the pole name beginning with a letter. Reference nodes within a submodel should be denoted by the same symbols as the reference nodes in system diagrams. They can, but need not be associated with pole labels.
- Those parameters of submodel parts that should be controllable from outside of the submodel (i.e. the submodel external parameters) denote by identifiers beginning with a letter. Specify the default numerical value of each such parameter should be an explicit equation.

Example. The following diagram of the submodel titled *Body in vertical plane* is stored in the BOB.DIA file. The body motion in the x - y rectangular coordinate system is modeled as the motion of the mass point A without any resistances. Note that m and g represent external parameters of the submodel.

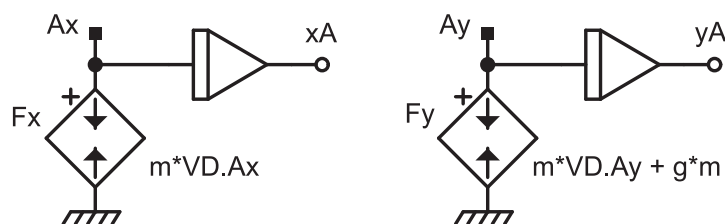


A submodel diagram can be also created by copying a system diagram part into the window for creating the submodel diagram and editing it graphically as described above.

16.1.2 Submodel equations

Dynamic behavior of submodels can be also characterized by systems of nonlinear algebro-differential equations. In such case submodel poles are formed by physical elements or blocks controlled by equations.

Example. Motion of the mass point A considered in the previous example can be described by the equations $F_x = m dA_x/dt$ and $F_y = m dA_y/dt + mg$. In the following diagram of the submodel BOBEQ these equations control sources of force, the poles of which form poles of the submodel.



16.2 Submodel text files

16.2.1 Submodel text files from diagrams

During any analysis, DYNAST does not process submodels characterized by their diagrams in the graphical form, but in the text form. For a submodel diagram stored in the *submodel.DIA* file, DYNAST generates automatically the *submodel.MOD* file in a text form. To see this file for a submodel diagram shown in the active diagram window, choose View Problem or Submodel from the View menu.

Example. For the submodel diagram BOB.DIA introduced in the first example DYNAST will generate the BOB.MOD file shown in the left column:

:: Body in vertical plane	:: Body in vertical plane
BOB	BOB :: mass point
Ax,	Ax, :: [m/s] x-velocity
Ay;	Ay/ :: [m/s] y-velocity
m = 1;	m = 1, :: [kg] point mass
g = 9.81;	g = 9.81; :: [m/s] acceleration g
I1 > @INT Ax, xA;	I1 > @INT Ax, xA;
I2 > @INT Ay, yA;	I2 > @INT Ay, yA;
m_x > C Ax = m;	m_x > C Ax = m;
m_y > C Ay = m;	m_y > C Ay = m;
Fg > J Ay = g*m;	Fg > J Ay = g*m;
EO@;	EO@;
	::xA [m] x-position
	::yA [m] y-position

A MOD file generated automatically for a submodel diagram should be slightly modified using the DYNAST text editor to make it compatible with the form specified below. An example of such a modified MOD file is shown in the right column above. Note, however, that the added comments right to the double semicolon characters :: are optional only as explained further.

:: description of modeled component	
<i>submodel</i>	:: submodel type
<i>pole -</i>	:: pole description
...	
<i>pole</i>	:: pole description
[/	
<i>parameter</i> [= value],	:: [unit] parameter description
<i>parameter</i> [= value],	:: [unit] parameter description
...	
<i>parameter</i> [= value];	:: [unit] parameter description]
...	
<i>dynamics</i>	
...	
EO@;	
:: <i>variable</i>	[unit] variable description
...	

submodel is a user-defined name of the submodel and – at the same time – an indication of the submodel type

pole is a user-defined name of a submodel pole identical with the name a submodel node (with the exception of the reference nodes). Pole names are mutually separated either by comma , or hyphen – characters.

parameter is a user-defined name of an external parameter the value of which can be controlled from outside of the submodel. The list of such parameters is separated from the list of poles by the slash character /.

value is a numerical constant or a symbolic expression specifying the default value of an external parameter, which applies if this value is not specified outside of the submodel. If the string = *value* is missing, the default value is zero.

dynamics represents statements specifying the submodel dynamics in terms of physical elements, blocks, submodels, or equations.

EO@ is the string closing the submodel description.

In addition, you can also specify in the MOD file values of initial conditions for the submodel solved variables. These initial values apply for the nonlinear numerical analysis of the system containing the submodel, unless other initial-condition values for the same variables are specified in the TR section of the system PRB file. The initial conditions are specified in the MOD files in the following form:

INIT *variable* = *expression* [, *variable* = *expression* ...];

variable is the name of a solved variable

expression is the numerical constant or a symbolic expression

16.2.2 Creating submodel dialogs

Comments placed to the right of the double semicolon characters :: in submodel MOD files are used for automatic generation dialogs which allow easy entering parameters or other submodel properties. Texts of some of the comments are displayed in tables or plots resulting from DYNAST analyses.

Table 16.1: Comments beginning with :: characters.

POSITION	TEXT OF COMMENTS
first row of MOD file	description of the modeled real component
next to submodel identifier	description of submodel type
next to pole identifier	description of pole
next to external parameter	description of external parameter
below EO@; statement	variable identifier [unit] description of variable

Example. The dialog generated automatically for the submodel file BOB.MOD is of the form

Submodel Properties

Teleso ve svisle rovine
hmotny bod bez odporu

☒ **Name:** Type:
bob1 bob.mod

☒ **Parameters:**

Parameters from a catalog:

Parameter	Value	Description
m	1	[kg] hmotnost bodu
g	9.81	[m/s ²] zrychleni g

10⁻¹⁵ 10⁻¹² 10⁻⁹ 10⁻⁶ 10⁻³ 10⁰ 10³ 10⁶ 10⁹ 10¹² | π Expression...

Example of
a submodel
dialog.

16.2.3 Creating submodel text from scratch

You can create the MOD text file for a new submodel directly without previous creating a diagram, of course. Using the DYNAST text editor follow the instructions for MOD file form given in the previous paragraphs. To open a new MOD file,

1. Choose New in the File menu.
2. Select Submodel text in the File type roll-down menu.
3. Enter the name of the file indicating also the submodel type in the File name field.
4. Indicate the modeled real component or phenomenon in the Submodel of field.
5. In the Template section, select Standard template unless you want to document the created submodel.
6. Click OK.

16.2.4 Submodel text from a problem text

You may find convenient this three-step procedure for developing and debugging MOD files for new submodels:

1. Develop a PRB file with the model of a system suitable for validating new submodels and containing their specification without the submodel formalism.

2. After debugging the PRB file encapsulate the submodel specifications in the following way:

```
*SYSTEM;
DEFMAC specification of submodel dynamics
EO@;
DEFMAC specification of submodel dynamics
EO@;
.....
specification of the rest of the validating system
specification of the system analysis
*END;
```

3. Debug now the newly created submodels together with the validating system (this option does not apply to DYNAST Server).

Note that the submodel lines stating in the PRB file must follow the `*SYSTEM;` statement, wh.

Texts of the debugged submodels preceded by the `DEFMAC` statements should be placed at the very beginning of the PRB file immediately behind the `*SYSTEM;` :

```
*SYSTEM;
DEFMAC specification of submodel dynamics
EO@;
DEFMAC specification of submodel dynamics
EO@;
.....
specification of system dynamics
specification of system analysis
*END;
```

Example. In this way, the text of the submodel BOBEQ.MOD can be debugged, for example, in such a PRB file:

```
*SYSTEM; :: Body in vertical plane
DEFMAC bobeq Ax, Ay/ m = 1, g = 9.81;
I1 > @INT Ax,xA; I2 > @INT Ay,yA;
m_x > C Ax = m; m_y > C Ay = m;
Fg > J Ay = g*m; EO@;
bobeql > @bobeql 1,2;
*TR; TR 0 .2; PRINT (501) bobeq1.xA, bobeq1.yA;
INIT bobeq1.Ax=1, bobeq1.Ay=1; RUN; *END;
```

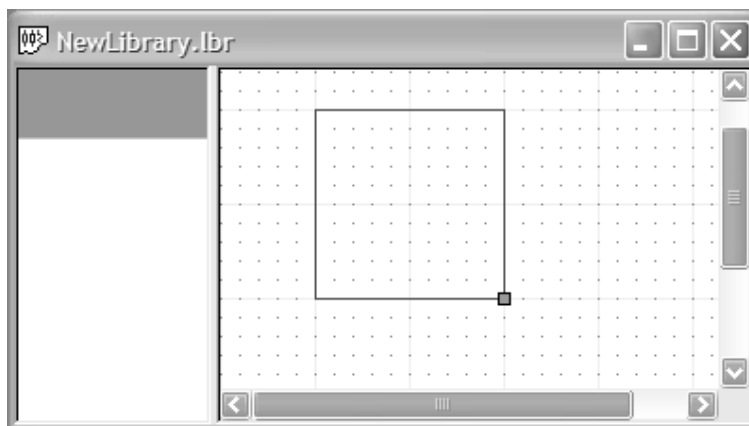
16.3 Libraries of submodel symbols

16.3.1 Creating a new library

Graphical symbols of submodels are organized in groups forming symbol libraries. Each of the libraries is stored in a LBR file. A new symbol library can be created in the following way:

1. From the File menu, choose New.
2. Select Symbol library in the File type roll-down menu.
3. In the File name text box, type the name of the symbol library you are creating.
4. In the Create in field, specify a folder for storing the library. Otherwise the library will be created in the main submodel folder.
5. Click OK.

The window of the symbol graphical editor displays for the created library. In the left part of the window, there is the empty list of symbols stored in the library. A symbol selected in this list appears in the right part of the window. If the library is still empty, the empty box bounding the symbol pattern to be created is shown there.

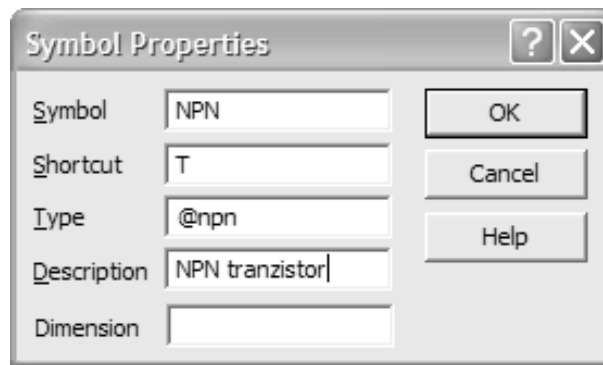


New
symbol
library.

16.3.2 Basic symbol properties

Before the pattern of a new symbol is created, the symbol basic properties should be specified:

1. Choose Symbol Properties in the Edit menu, or double click the left part of the window. The Symbol Properties dialog displays.
2. Type the symbol name in the Symbol text box of the dialog.
3. In the Shortcut text box, type the symbol shortcut that will be used as the default name of all symbol instances when they are placed in a system diagram.
4. In the Type text box, type the name of the related submodel MOD file.
5. Type a short symbol description in the Description text box.



Specification
of symbol
properties

16.3.3 Creating symbol patterns

In the right part of its window, the symbol editor displays the red bounding box of the symbol pattern. Pins will be placed on the symbol outside of this region, touching the symbol bounding box border. To change the size and shape of the bounding box, drag the green point from the right bottom corner of the border to the required intersection of two lines in the underlying grid.

The tools that are at your disposal for creating a symbol pattern are listed in Table 16.2. You can choose any of the tools either from the Place menu or from the toolbar. Using the tools, create the symbol within the chosen bounding box. Each of the drawing tools remains active until the Esc key or the right mouse button is pressed, or another tool is chosen.

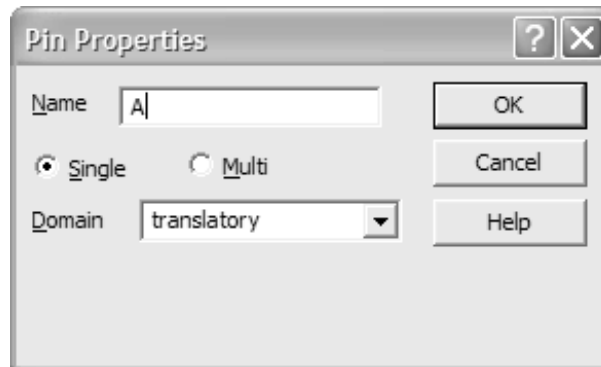
Table 16.2: Tools for graphical editing symbols.

ICON	PLACE MENU COMMAND	DESCRIPTION
	Polyline	Polyline chain defined by clicking its braking points.
	Polygon	Closed polygonal chain of line segments defined by clicking its braking points.
	Rectangle	Rectangle defined by clicking its two diametral corners.
	Circle	Circle defined by clicking first its center and then a point of its curve.
		Filled versions of the above patterns.
	Arc	Arc defined by clicking first its center and then its endpoints. The endpoints will be interconnected in the clockwise direction.
	Text	text string is placed by clicking a point within the symbol pattern.


To insert a text into a symbol pattern, double click the string **text** placed in it (assuming no other graphical tools is active). In the displayed dialog, choose either Text rotates with the symbol, or Text keeps its orientation, i.e. the text rotates with the symbol.

16.3.4 Placing symbol pins

Symbol pins allow the symbol interconnection with the pins of other symbols in system diagrams. The pins emanate from the grounding boxes of symbol patterns. They are placed along the lines forming a grid in the graphical editor.



Entering properties of single pin.

To associate a pin with a symbol, choose Pin from the Place menu or click the toolbar icon . Move the pin by the mouse along the symbol border and click the desired place. The pin length can be set zero without any impact on its functionality.

To specify pin properties,

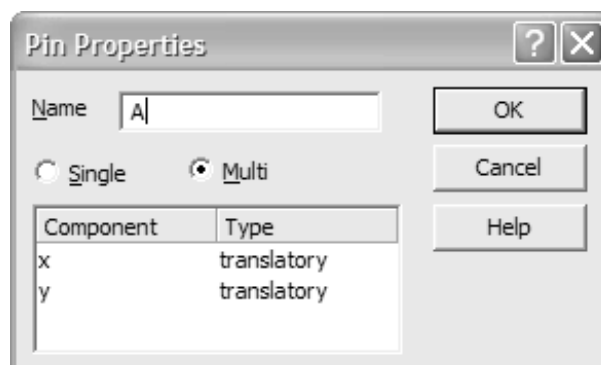
1. Click successively each pin and, in the displayed dialog Pin Properties, enter the pin name.
2. Choose either Single or Multi depending whether the pin is single or multiple.
 - In the former case, only the pin energy domain should be specified: generic, electric, magnetic, thermal, fluid, rectilinear or rotary. This information is used to check the mutual compatibility of interconnected pins.
 - In the latter case, specify names and energy domains of all multiple pin components.

The name of each single pin should be identical with the name of the corresponding submodel pole specified in the submodel MOD file. A multiple pin represents several poles. In this case, names of submodel poles corresponding to the individual components of a multiple pin consists of the multiple-pin name combined with the name of the multiple-pin components related to the poles:

pincomponent

pin is a name of the multiple pin.




component is a name of the multiple pin component.



Entering properties of multiple pin.

Example. The dialog shows specification of the multiple pin A with two components x and y representing poles A_x and A_y of the submodel BLOB mentioned earlier.

16.3.5 Editing symbol patterns

Objects forming a symbol pattern can be edited after pressing the Esc key, or mouse right-clicking, or clicking the  toolbar icon. All changes can be canceled or restored by choosing Undo or Redo from the Edit menu. You can also click either the  or  toolbar icon.

Object selection. Click the right part of the library outside of any object. Then either click successively the selected objects while holding the Ctrl key. You can also inclose the objects into a rectangular set by dragging the mouse diagonally across the area with the objects.


Object copying. Hold down the Ctrl key and drag the copy of the selected objects to a new position. You can also choose subsequently the Copy and Paste commands from the Edit menu. This allows also copying pattern objects to another symbol or in the same or different library.


Object editing. You can change the size or shape of the selected objects by dragging their handles to the required positions.

Text editing. Double click the **text** string in a symbol pattern to edit it. Similarly, double click the a pin symbol to edit the pin description.

16.3.6 Editing symbol libraries

You can edit a symbol after selecting it in the left part of the symbol editor.

Symbol adding. Choose New Symbol from the Edit menu, or click the  icon in the symbol editor toolbar.

Symbol duplicating. Choose Duplicate Symbol from the Edit menu, or click the  icon in the symbol editor toolbar.

Symbol reordering Change the order of symbols in the library by dragging the symbols in left part of the symbol editor.

Symbol removal. Select the symbol to be deleted and press the Del key.

Symbol copying. To copy a symbol from one library to a another one, select the text of the symbol in the LBR file of the first library and paste it into the LBR file of the second library using a text editor.

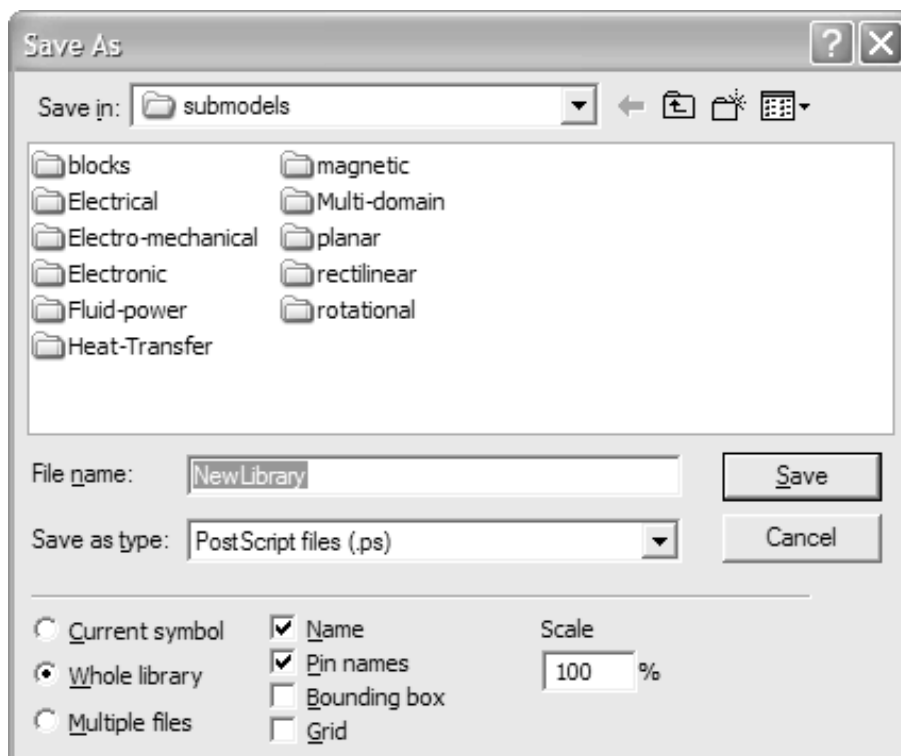
Also these changes can be canceled or restored by choosing Undo or Redo from the Edit menu. You can also click either the  or  toolbar icon.

If you want to place a changed symbol into a system diagram before closing DYNAST, choose Refresh Libraries from the Edit menu.

16.3.7 Exporting symbol libraries

To export a symbol library opened in the symbol graphical editor, choose Export to PostScript from the File menu. You can then choose from the displayed dialog

- Current symbol to export the symbol pattern to an EPS file
- Whole library to export the complete symbol library to a PS file
- Multiple files to export all symbols from the library to independent EPS files



Exporting
symbol
library to
PostScript

You can also decide which symbol details will be exported besides their patterns. In the Scale box, you can set the scale of the exported symbol patterns. If the scale of 100% is set, the line spacing in the exported underlying grid is 5 mm.

16.4 Organization of submodel files

As mentioned earlier, information related to individual submodels can be stored in files with the following extensions:

MOD files specify submodel dynamics and external interactions in a textual form

LBR files store libraries of graphical symbols of the individual submodels

DIA files represent submodel dynamics and external interactions in a graphical form (optional)

PDF files with texts documenting the individual submodels

In the installation of the working environment DYNAST Shell, all files related to submodels are stored in the folder SUBMODELS. The submodel files are organized in subfolders and subsubfolders of the SUBMODELS folder in a hierarchical way. Each of the subfolders is assigned for a specific class of submodels (Electric, Electronic, Fluid, etc.).

The organization of submodel files in the SUBMODELS folder is following:

SUBMODELS folder contains

- subfolders – each of them assigned to a class of submodels
- ASCII file INDEX. specifying order and titles of submodel classes

Subfolders contain



- LBR files with symbol libraries related to the submodel class of the subfolder
- subsubfolders – one for each symbol library
- ASCII file INDEX. specifying order and titles of submodel groups related to symbol libraries

Subsubfolders contain

- MOD files of individual submodels – all of them from a group related to one of the symbol libraries
- DIA files with diagrams related to MOD files of individual submodels (optional)
- PDF files with texts documenting the individual submodels
- ASCII file INDEX. specifying order and titles of submodels in a group related to a symbol library

If you would like to store your submodel files in different folders, you should associate them with DYNAST Shell (see Chapter 2).

The list submodels in the associated folders displays if you choose

- either List of Submodels from the View menu, or clicking the  toolbar icon,
- or Place Part from the Place menu, or clicking the  toolbar icon when the window for creating diagrams is active

Chapter 17

Modeling toolbox for MATLAB

17.1 DYNAST & MATLAB in control design

DYNAST can be easily used as a *modeling toolbox for MATLAB*. While MATLAB is well suited to control design, DYNAST is capable of automated equation formulation even for very realistic models of real systems. You can combine these two programs to exploit advantages of both. Notably, you can implement and analyze a nonlinear model of the plant to be controlled in DYNAST. Then, for example, you may ask DYNAST to linearize the model, compute transfer-function poles and zeros of the plant model, and export them for the plant-control synthesis to MATLAB. Finally, to verify the complete controlled system you can use DYNAST again after augmenting the plant model by the resulting control configuration. During this design phase, you may use DYNAST to consider also plant nonlinearities as well as the non-ideal features of the controllers and sensors in various operation regimes of the control system. If the designed control is digital, you may verify it by interconnecting DYNAST with Simulink so that these two packages can communicate with each other at each time step.

How to do it, both in the case of analog and digital control, is described in this chapter. Even the DYNAST sitting on our server (or any other server) can communicate with MATLAB or Simulink installed on your own computer across the Internet. See the instructions and examples linked from <http://virtual.cvut.cz/dyn/>

17.2 Exporting transfer functions to MATLAB

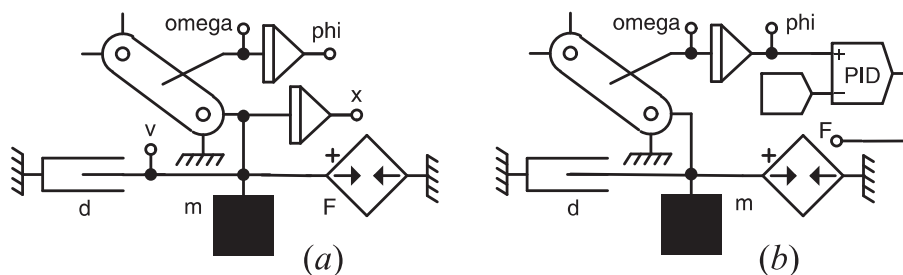
Transfer-function coefficients resulting from semisymbolic analysis in DYNAST can be easily transported to MATLAB or other programs. To export to MATLAB

1. Following instructions in Chapter ?? open the window Semisymbolic Results with transfer functions of the dynamic system.
2. Choose the transfer function you want to export (or if you wish to export all the functions choose their folder) and click the Copy to Clipboard button.
3. From the MATLAB menu Edit choose the command Paste.
4. Verify the export by inserting the variable identifiers to the command line of MATLAB and press the Enter key.

Example. Let us assume, that you want to design analog control of the carriage with inverse pendulum the model of which is shown in Fig. *a*. The carriage drive should be controlled in such a way that the pendulum remains in the prescribed position. The carriage inertia and damping related to its horizontal motion is represented by the inductor m and damper d . The action of the carriage drive is modeled by the source of force F . The submodel ROD from the DYNAST library was exploited to model the pendulum motion in the vertical plane. Evaluation of the carriage position x from its velocity v , and the angular displacement ϕ of the pendulum from its angular velocity ω is provided by integrators.

After exporting coefficients of the system transfer function $H(s) = \phi(s)/F(s)$ to the MATLAB environment you can design a PID control for the system, for example. Fig. *b* shows the system model with the controller regulating the source of force F .

For more details see Tilbury and B. Messner: *Control Tutorials for Matlab*, <http://www.engin.umich.edu/gr>



V prípade nelineárnych sústav, ktoré majú bť zeny, DYNAST podstatne usnadňuje vyetovň jejich detailnho chovň bez zptň vazby. Umouje nalzt jejich klidov pracovň bod pomoc nelineárn statick analyzy, v tomto bod je linearizovat a urit jejich penosov funkce i odezvy na poten stav. Po dokonen nvrhu zen DYNAST dovoluje dkladn oven cel zptnovazebň soustav y pi respektovň nelinearit a dalch neidelňch vlastnost jak zenho objektu, tak i dcch prvk.

17.3 Controlling model in DYNAST by Simulink

Using a controlling structure represented by a block diagram implemented in Simulink you can control a plant model implemented in DYNAST. In the Simulink block diagram, the DYNAST plant model is represented by a block called S-function. This block provides communication between Simulink and DYNAST.

17.3.1 Preparing the plant model in DYNAST

Required arrangement of PRB file DYNAST The DYNAST problem text PRB file with the plant model to be controlled should be arranged in the following way:

- The PRB file must specify the numerical transient analysis of the plant model (it must be the first analysis if several analyses are specified there).
- The analysis time interval must be as long as the time interval specified for the simulation in Simulink at least.
- At the end after the `*END;` command the PRB file must include the following statement in the form of a comment placed in a single line:

```
: MATLAB interface spec: ' input,input, ...';' output,output, ...'
```

input is the identifier of the parameter of an E or J source, independent BS block, or of an equation parameter

output is the identifier of the desired variable of the analyzed model

Example. In the case of the inverted-pendulum problem this statement should be of the form

: MATLAB interface spec: 'F';'phi,omega,x,v'

17.3.2 Required setup of MATLAB

Konfiguraci MATLABu pro soubnou simulaci DYNASTu se Simulinkem si provete v nasledujicich krocich:

1. Activate program MATLAB.
2. Apply the command Set path from the File menu.
3. Choose Add Folder.
4. Nalistujte sloku uvedenou ne.
5. Click the Save button for saving the setup.
6. Click the Close button to close the window.

Nalistovan sloka m podobu

folder\matlab\ toolbox\DYNAST

folder is the name of the folder in which DYNAST and its environment is stored on your computer (usually C:\Program Files\DYNAST)

toolbox is the character string toolbox6 for MATLAB version 6, or toolbox7 for MATLAB version 7

17.3.3 Preparing the control diagram in Simulink

In Simulink, set up the block diagram of the control system with an S-function block from the Simulink Nonlinear library representing the plant model. Then specify the following parameters for the S-function block:

V Simulinku je poteba vytvoit blokov schma dc struktury s blokem S-Function predstavujcm model zen soustavy implementovan v DYNASTu. Uveden blok si zadte tmto postupem:

1. Set up the block diagram of the complete control system
2. Choose the block of type S-Function from the Nonlinear library or User-Defined Functions.
3. Type to the S-function name box the string of characters dynPlantL
4. Type to the S-function parameters box the

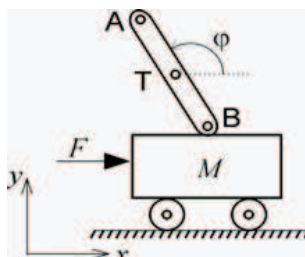
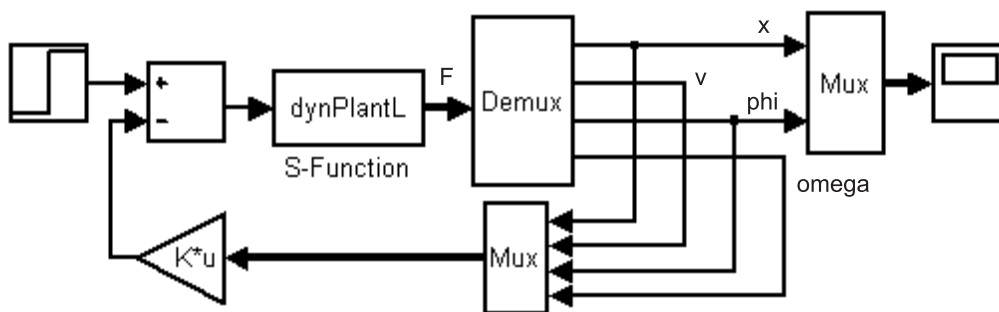
'path', sampling

path is the complete path to the PRB file with the controlled plant model in the DYNAST environment. You can use the string %DYNAST% to denote the folder with DYNAST Shell.

sampling sets the time interval (in seconds) in which the data between DYNAST and Simulink will be exchanged during the common simulation.

As soon as simulation in Simulink is done, it automatically sends the data to DYNAST with the model and the system. Simulink and DYNAST then save the data and use it for the next simulation.

Example. The figure shows the block diagram for the digital state control of the inverted-pendulum plant in the Simulink environment. The `dynPlantL` block represents the dynamic model shown in Fig. 17.1 above. Model of the cart with a pendulum, implemented in DYNAST, is in the block diagram represented by the S-Function block.



Bibliography

- [1] Rubner-Petersen, T.: *Nonlinear Analysis Program NAP3* (an unfinished project). DTH, Lyngby 1980
- [2] Mann, H.: *Computer applications in electrical engineering design* (in Czech). SNTL Publishing House, Prague 1984
- [3] Oliva Z.: *Some algorithms for electronic circuit analysis* (in Czech). PhD. thesis, Czech Technical University, Prague 1986
- [4] Mann, H.: *Multipoles, multiports and operational blocks*. Proc. European Conf. Circuit Theory and Design, London 1974
- [5] Mann, H.: *Analysis of combined circuit-block diagrams*. Proc. Int. Symp. on Circuits and Systems ISCAS IEEE, Rome 1982, 639-642
- [6] Rubner-Petersen, T.: *ALGDIF – a FORTRAN IV subroutine for solution and perturbed solutions of algebraic-differential equations*. Research Report, DTH, Lyngby 1979
- [7] Gear, C.W.: *Numerical initial value problems in ordinary differential equations*. Prentice-Hall, Englewood Cliffs, N.J. 1971
- [8] Brigham, E.O.: *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, N.J. 1974
- [9] Mann, H.: *An algorithm for the formulation of state-space equations*. Proc. 1979 Int. Symp. on Circuits and Systems ISCAS IEEE, Tokyo 1979, 161-162
- [10] Rubner-Petersen, T.: *SFORM1 and SFORM2 – two FORTRAN IV subroutines for sparse matrix transformation of the general eigenproblem to standard form*. Research Report IT-41, DTH, Lyngby 1979
- [11] Mann, H. et al.: *Computer-aided design of dynamic systems*. CSVTS House of Engineering, Prague 1986
- [12] Mann, H.: *Theory of mechanical systems II*. Textbook, Technical University, Brno 1990